THE UNIVERSITY
of EDINBURGH

Evaluation-Efficient Multidimensional Numerical Integration

Greg Tumolo

Doctor of Philosophy
2018

Declaration

I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where stated otherwise by reference or acknowledgment, the work presented is entirely my own.

_____

Acknowledgements

I acknowledge my supervisor, Dr. J. Mark Bull, for his help and, especially, his belief in me. I also acknowledge my parents, Frank & Maria Tumolo, for their support, without which this work would have been impossible.

Lay Summary

Consider a lake. Suppose you know the area of the surface of the lake and want to numerically estimate the volume of the lake. You could evaluate the depth of the lake at random sites on the surface and, then, estimate the volume by the area times the average depth. If you were to do so, you would be applying the simplest method for the numerical estimation of the integral (which is analogous to the volume) of a multidimensional integrand (which is analogous to the lake), which is **multidimensional numerical integration**.

Assume an evaluation of the depth of the lake is expensive (in units of time or other precious resource). If you were to apply a less simple method for the numerical estimation of the volume of the lake, you could achieve the same accuracy at a lower cost by evaluating at fewer (not necessarily random) sites. If you were to do so, you would be applying a more **evaluation-efficient** method.

In this thesis, VoroInt (Voronoi Integrator), which is a novel method for **evaluation-efficient multidimensional numerical integration**, is presented and compared to other methods. VoroInt estimates the integral of a multidimensional integrand by the sum of, for each site, the area of the Voronoi region of the site times the value of the integrand at the site. The Voronoi region of a site is the collection of the potential sites that are nearer or as near to the site than they are to any other site. Continuing the lake analogy, VoroInt estimates the volume of the lake by the sum of, for each site, the area of the Voronoi region (of the surface of the lake) of the site times the depth at the site. VoroInt can estimate the integral using any sites. However, for evaluation efficiency, VoroInt randomly selects each site from the Voronoi region for which the term in the sum is probably the most erroneous. This thesis was motivated by an application of multidimensional numerical integration in cosmology for which the collective evaluations of the integrand are prohibitively expensive.

Abstract

Pre-existing methods for multidimensional numerical integration tend to use many more evaluations of the integrand than necessary. That is, pre-existing methods are evaluation inefficient. In some applications of multidimensional numerical integration, such as cosmological model selection, each evaluation is expensive. Consequently, multidimensional numerical integration by way of any pre-existing method can be prohibitively expensive. An evaluation-efficient method for multidimensional numerical integration could enable such applications.

Presented in this thesis is VoroInt (Voronoi Integrator), which is a novel Monte Carlo method that uses a Voronoi decomposition of the integration domain into regions about sample sites. The approximate integral returned by the method is a Riemann sum of the value at each site weighted by the size (that is, area, volume, or hyper-volume) of the Voronoi region that contains the site. The error estimate returned by the method is the sum of, for each region, the error estimate for the approximate integral over the region. The error estimate for a region is the average deviation of the values at the sites in the regions that are adjacent to the region from the value at the site in the region times the size of the region. A similar formula in which the average deviation is replaced by the absolute maximal deviation (because the latter is more robust than the former) is used by the method to prioritize each region for further sampling. Also presented are comparisons of the number of evaluations used by and the accuracy of the values returned by pre-existing methods and the novel method for various test integrands, which show that the evaluation efficiency *and* the accuracy of the novel method are better than those of pre-existing methods. The improvements come from the effective sampling and error estimation by the method. Effective sampling enables the method to efficiently find features of the integrand that significantly contribute to the integral. Effective error estimation enables the method to well estimate the accuracy of the approximate integral.

Contents

1 Introduction

Pre-existing methods for multidimensional numerical integration tend to use many more evaluations of the integrand than necessary to numerically integrate the integrand within some tolerance. That is, pre-existing methods are evaluation inefficient. In some applications of multidimensional numerical integration, such as cosmological model selection ([1]), the motivator of this thesis, pre-existing methods use hundreds of thousands of evaluations, each of which is expensive (that is, costs more than a second of CPU time), which can cost more than a week of CPU time. Consequently, multidimensional numerical integration by way of any pre-existing method can be prohibitively expensive. An evaluation-efficient method for multidimensional numerical integration could enable integrals that were prohibitively expensive to be computed in an affordable amount of CPU time. In cosmological model selection, where $\mathscr{L}$ is the likelihood of a model, the relevant integral is the Bayesian evidence. From [1]:

"The Bayesian evidence [is] defined as

$$E = \int \mathscr{L}(\theta) \Pr(\theta) d\theta.$$

Here θ is the vector of parameters of the model, and $\Pr(\theta)$ is the prior distribution of those parameters before the data were obtained. The prior is an essential part of the definition of a model, upon which the evidence will ultimately depend, and might for instance be a set of ranges within which parameters are assumed to be uniformly distributed.

The evidence of a model is thus the average likelihood of the model in the prior. [I]t does not focus on the best-fitting parameters of the model, but rather asks "of all the parameter values you thought were viable before the data came along, how well on average did they fit the data?". Literally, it is the likelihood of the model given the data. Given Bayes' theorem

$$P(M \mid D) = \frac{P(D \mid M) P(M)}{P(D)},$$

where M is the model, D is the data, and the vertical bar is read as 'given', the evidence $E \equiv P(D \mid M)$ updates the prior model probability $P(M)$ to the posterior model probability $P(M \mid D)$, i.e. the probability of the model given the data.

The evidence rewards predictability of models, provided they give a good fit

to the data, and hence gives an axiomatic realization of Occam's razor. A model with little parameter freedom is likely to fit data over much of its parameter space, whereas a model which could match pretty much any data that might have cropped up will give a better fit to the actual data but only in a small region of its larger parameter space, pulling the average likelihood down.

The evidence is also known as the marginalized likelihood or, more accurately, the model likelihood. The ratio of evidences for two models is known as the Bayes factor."

When $P(D)$ is unknown and, therefore, $P(M \mid D)$ cannot be computed, a Bayes factor is used to compare two models M_1, M_2 for which (by the above definition) $P(M_1) = P(M_2)$ because

$$\frac{P(M_1 \mid D)}{P(M_2 \mid D)} = \frac{\frac{P(D \mid M_1)P(M_1)}{P(D)}}{\frac{P(D \mid M_2)P(M_2)}{P(D)}} = \frac{P(D \mid M_1)}{P(D \mid M_2)} = \frac{E_1}{E_2}$$

does not depend on $P(D)$.

Methods for multidimensional numerical integration that evaluate the integrand at grid points, such as the methods discussed in [2], are not considered in this thesis because the evaluation efficiencies of such methods suffer from "the curse of dimensionality". That is, the numbers of evaluations of the integrand used by such methods to numerically integrate the integrand within some tolerance dramatically increase as the dimensionality of the integrand increases. Furthermore, such methods do not robustly explore the integration domain for valuable features of the integrand. Rather, Monte Carlo methods ([3]), which evaluate the integrand at random points, are considered because they dispel the curse and robustly explore the integration domain. Some Monte Carlo methods also exploit their nondeterminism (up to the sequence of random numbers generated by their source of randomness) to increase or quantify the confidence in their result by combining intermediate results from multiple executions. Markov Chain Monte Carlo (MCMC) methods, such as the Metropolis-Hastings algorithm ([4-5]), are excluded because they often converge extremely slowly and,

therefore, are often extremely inefficient.

The pre-existing Monte Carlo methods for multidimensional numerical integration considered in this thesis are the standard methods and the most-frequently-used method in cosmological model selection and its basis. The standard methods are PLAIN, which is the traditional method, VEGAS ([11-12]), which uses adaptive importance sampling and stratified sampling, and MISER ([13]), which uses recursive stratified sampling. The most-frequently-used method in cosmological model selection, which is based on nested sampling (Nest ([21])), is MultiNest ([22]).

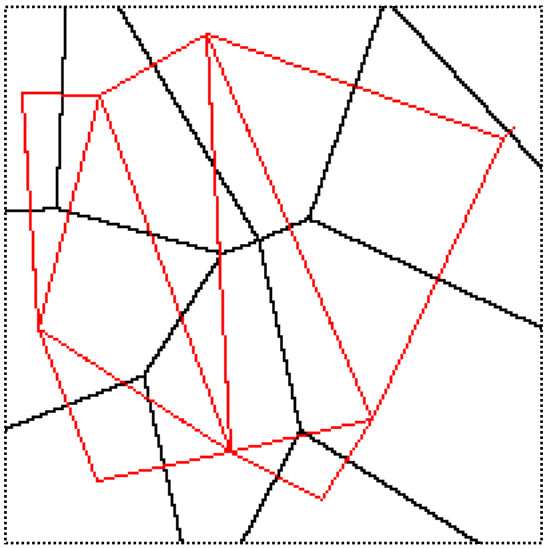


Figure 1-1: A (black) Voronoi decomposition and (red) Delaunay triangulation

The novel Monte Carlo method for multidimensional numerical integration considered in this thesis uses computational geometry. It directly uses Voronoi decompositions and Delaunay triangulations, as in Figure 1-1 (above), and indirectly uses convex hulls, as in Figure 1-2 (below). The Voronoi decomposition of a domain about special points in the domain is a decomposition of the domain into subdomains such that each subdomain contains a special point and every point in the domain for which the distance from the point to the special point is less than or equal to the distance from the point to any other special point; the Delaunay triangulation of the special points is a graph of their adjacency by way of their subdomains. In the

figures, the endpoints of the red line segments are the special points, the red line segments are the edges of the Delaunay triangulation of the special points, and the black line segments are the boundaries of the subdomains of the Voronoi decomposition of the domain about the special points. The convex hull of the special points is the smallest convex set that contains the special points. The boundary of a convex hull is described by special points. In the below figure, the blue line segments are the boundaries of the convex hull of the special points.

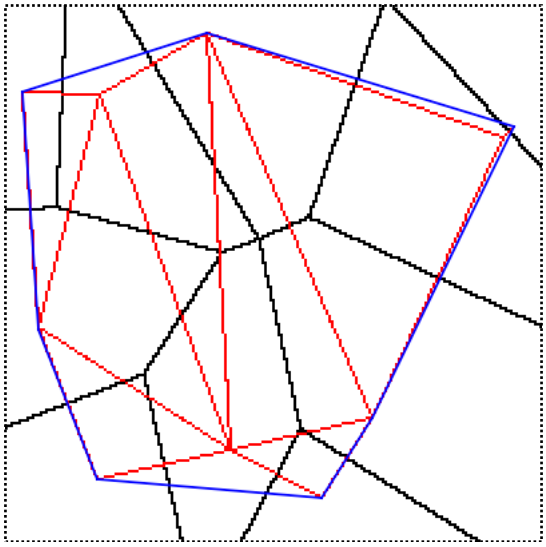


Figure 1-2: The (blue) boundary of a convex hull

Presented in this thesis is VoroInt (Voronoi Integrator), which is a novel Monte Carlo method for multidimensional numerical integration that uses a Voronoi decomposition of the integration domain about random samples (evaluations) of the integrand. With each sample, the Voronoi decomposition associates a subdomain and a natural set of neighbors, which are the samples associated with the adjacent subdomains. The (hyper-)area of a subdomain is also computed by way of computational geometry. The (hyper-)area of each subdomain is used with the value of its associated sample to compute the partial integral over the subdomain and with the values of the neighbors of the sample to estimate the error in the partial integral and to prioritize the subdomain for random sampling by the method. A subdomain is randomly sampled by rejection sampling from its bounding

box. The use of a Voronoi decomposition for numerical integration is not novel; however, the use of a Voronoi decomposition for adaptive numerical integration is novel. Adaptive numerical integration proceeds by refining the subdomain with the greatest error estimate. Therefore, the ability to estimate the errors in the partial integrals enables VoroInt to adapt. Superficially, the algorithm for VoroInt is like the approach to adaptive numerical integration introduced in [6], which entails iteratively bisecting the rectangular subdomain with the current greatest (local) error estimate and applying an integration rule to each half of the subdomain until the global error estimate is tolerable. An integration rule specifies a point or points (in the subdomain to which the rule is applied) at which to evaluate the integrand for integration and error estimation. When the integration is multidimensional, the approach demands the dimension of each bisection be chosen; a bad choice does not reduce the global error estimate as much as a better choice would have done and, therefore, could cause unnecessary evaluations. VoroInt avoids the difficulty of choosing the best bisection dimension, but modifies multiple subdomains per iteration—the subdomain with the current greatest error estimate and the other subdomains adjacent to the new subdomain. A means to select the other adjacent subdomains without excessive computational cost (in either time or memory) is the most significant contribution of this thesis.

VoroInt relies on Qhull ([7]), which is a popular computational geometry package, to compute Voronoi decompositions and areas. Qhull computes decompositions by transforming the samples to a space with an extra dimension, computing the convex hull of the transformed samples using the Quickhull algorithm for convex hulls ([8]), and applying the inverse transform to the convex hull as in [9]. Quickhull is a divide-and-conquer algorithm that is similar to Quicksort (whence its name came). Qhull computes areas by decomposing each subdomain into simplexes based on each facet of the subdomain that contain the centroid of the subdomain and the centroid of the facet.

Also presented in this thesis are comparisons of the number of evaluations used by and the accuracy of the values returned by pre-existing methods and VoroInt for various test integrands with properties typical of integrands in cosmological model selection. The comparisons show that the evaluation efficiency *and* the accuracy of VoroInt usually are better than those of pre-existing methods. The improvements come from effective sampling and error estimation by the method. Effective sampling enables the method to efficiently find features of the integrand that significantly contribute to the integral. Effective error estimation enables the method to well estimate the accuracy of the approximate integral, which enables the method to terminate maturely.

In the rest of this chapter, the mathematical terminology used in the rest of this thesis is defined (in Section 1.1), the integrand used to qualitatively demonstrate the operation of each method described in this thesis is introduced (in Section 1.2), and an overview of subsequent chapters is given (in Section 1.3).

1.1 Terminology

In this section, the mathematical terminology used in the rest of this thesis is defined.

1.1.1 Dimensionality

The dimensionality of the integration is:

$$2 \leq d \leq 6$$

d is an integer between 2 and 6 because such is necessary for the applications that are targeted by this thesis.

1.1.2 Bounds

The lower and upper bounds of the integration are:

$$\mathbf{l}, \mathbf{u} \in \mathbf{R}^d$$

1.1.3 Domain

The domain of the integration is:

$$D = [l_1, u_1] \times [l_2, u_2] \times \cdots \times [l_d, u_d] \subseteq \mathbf{R}^d$$

The (hyper-)area of D is:

$$A_D = \prod_{i=1}^{d} \overline{[l_i, u_i]}$$

1.1.4 Integrand

The integrand (function) restricted to D is:

$$f|_D : D \to \mathbf{R}$$

1.1.5 Point

A point in D is:

$$\mathbf{x} \in D$$

1.1.6 Integral

The (definite) integral of f over D with respect to $\mathbf{x}$ is:

$$I = \int_D f(\mathbf{x})\, d\mathbf{x} = \int_{l_1}^{u_1} \int_{l_2}^{u_2} \cdots \int_{l_d}^{u_d} f(x_1, x_2, \cdots, x_d)\, dx_1\, dx_2 \cdots dx_d$$

Note the multiple integral, whence comes the challenge.

1.1.7 Sample

A sample (that, is a point-value pair) is:

$$(\mathbf{x}, f(\mathbf{x}))$$

1.1.8 Weight

The method-dependent weight of a sample is:

$$w \in \mathbf{R}^+$$

1.1.9 Approximate Integral

The approximation of I using n samples and their weights is:

$$\tilde{I} = \sum_{i=1}^{n} w_i f(\mathbf{x}_i) \cong I$$

$\tilde{I}$ is a weighted sum of sample values.

1.1.10 Error

The error in $\tilde{I}$ is:

$$E = \tilde{I} - I$$

E is usually unknown because I is usually unknown.

1.1.11 Approximate Error

The method-dependent approximation of E is:

$$\tilde{E} \cong E$$

1.2 Demonstration Function

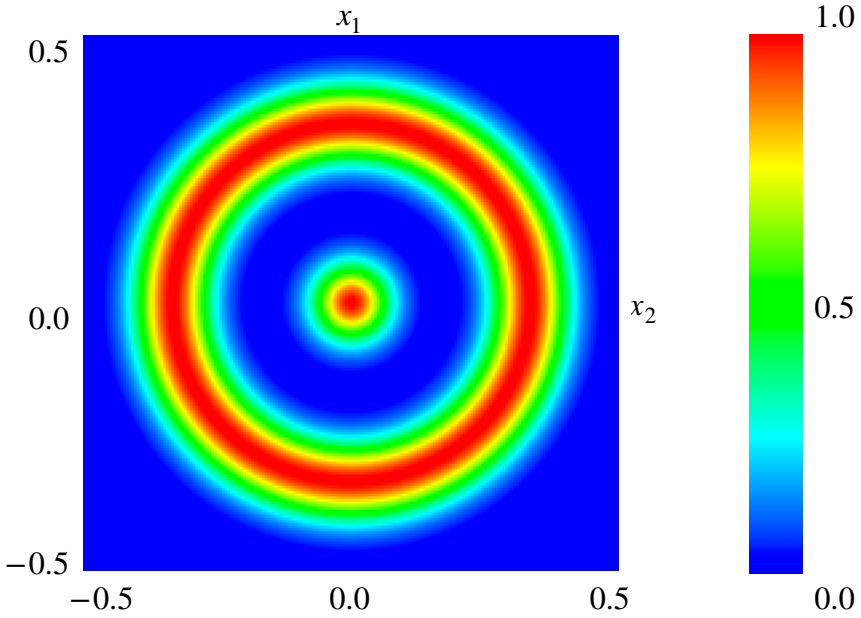


Figure 1.2-1: The demonstration function

In cosmological model selection, integrands are nonnegative and possibly multimodal (that is, they may have multiple features, which may be difficult to simultaneously detect), spiky (that is, they may have a narrow but valuable feature, which may be difficult to detect or resolve because of the high gradient of its edges), or degenerate (that is, they may have a curving feature, which may be difficult to resolve because it varies in multiple directions simultaneously). The function graphed in Figure 1.2-1 (above) on $D = [-0.5,0.5]^2$ with a colormap from $[0.0,1.0]$ (the codomain of the function) to spectral colors between blue and red is used to demonstrate the operation (for the same random seed) of the methods described in this thesis because the function is nonnegative, multimodal, spiky, and degenerate.

1.3 Overview

In Chapter 2, the standard methods for (multidimensional) Monte Carlo integration, which are PLAIN, VEGAS, and MISER, are described. In Chapter 3, nested sampling (Nest) and the most-frequently-used (Monte Carlo) method for (multidimensional) numerical integration in cosmological model selection, which is based on nested sampling, are described. In Chapter 4, VoroInt, which is a novel Monte Carlo method for multidimensional numerical integration that uses a Voronoi decomposition of the integration domain about random samples (evaluations) of the integrand, is presented. In Chapter 5, comparisons of the number of evaluations used and the accuracy of the values returned by the methods described in Chapters 2-3 and the method presented in Chapter 4 for various test integrands relevant to cosmological model selection are presented. In Chapter 6, this thesis is concluded with discussion of the results presented in Chapter 5 and suggestions for possible improvements to and applications of the method presented in Chapter 4.

2 Monte Carlo Integration

The three standard methods for (multidimensional) Monte Carlo integration, PLAIN, VEGAS, and MISER, use importance sampling, adaptive importance sampling or stratified sampling, and recursive stratified sampling.

Importance sampling is a technique to efficiently compute $\tilde{I}$ using n samples (indexed by i) distributed according to some distribution function $g|_D : D \to \mathbf{R}^+$ with weights:

$$w_i = \frac{\int_D g(\mathbf{x})\, d\mathbf{x}}{n} \frac{1}{g(\mathbf{x}_i)}$$

The weights of the samples are fractions of the integral of g over D that are inversely proportional to the asymptotic density of the samples. That is, w_i is small/large when $g(\mathbf{x}_i)$ is large/small to compensate for many/few samples in regions of high/low density. The goal of the technique is to concentrate samples in regions of importance (that is, high absolute value) by appropriately choosing g. However, doing so requires knowledge of the shape of $|f|$, which is usually unknown. **Adaptive importance sampling** is a technique to (iteratively) adapt g towards $|f|$ using samples distributed according to g and to use g for importance sampling.

Stratified sampling (in the context of numerical integration) is a technique to efficiently compute $\tilde{I}$ by decomposing D into strata (that is, subdomains) and sampling the stratum with the highest approximate error. $\tilde{I}$ is computed by summing the approximate integrals over the strata. **Recursive stratified sampling** is a technique to efficiently compute $\tilde{I}$ by recursively decomposing a stratum, beginning with D, into strata with a lower total approximate error than the approximate error in the stratum.

Described in the rest of this chapter are the methods PLAIN (in Section 2.1), VEGAS (in Section 2.2), and MISER (in Section 2.3) as implemented in the GNU Scientific Library (GSL) ([10]).

2.1 PLAIN Monte Carlo

In importance sampling, if $g \equiv 1$ (that is, g were the uniform distribution), then:

$$w_i = \frac{\int_D 1 \, d\mathbf{x}}{n} \frac{1}{1} = \frac{A_D}{n} \Rightarrow \tilde{I} = \sum_{i=1}^{n} \frac{A_D}{n} f(\mathbf{x}_i)$$

Note that w_i is constant with respect to i and that $n w_i = A_D$.

Consider $\tilde{I}$ written thus:

$$\tilde{I} = \frac{\sum_{i=1}^{n} A_D f(\mathbf{x}_i)}{n}$$

Clearly, $\tilde{I}$ is the mean of n one-point approximate integrals. The standard error in $\tilde{I}$ is:

$$\tilde{E} = \sqrt{\frac{\sum_{i=1}^{n} (A_D f(\mathbf{x}_i) - \tilde{I})^2}{n(n-1)}}$$

PLAIN uses $g \equiv 1$ and is the limiting case of importance sampling.

2.1.1 Demonstration

$10{,}000$ samples of the demonstration function by PLAIN look like those in Figure 2.1.1-1 (below). Note that both features are resolved but the unimportant (that is, low-valued or blue) regions, including the regions near the boundaries of the domain, are sampled as frequently as the important (that is, high-valued or red) regions. Because PLAIN samples indiscriminately, it is inefficient.

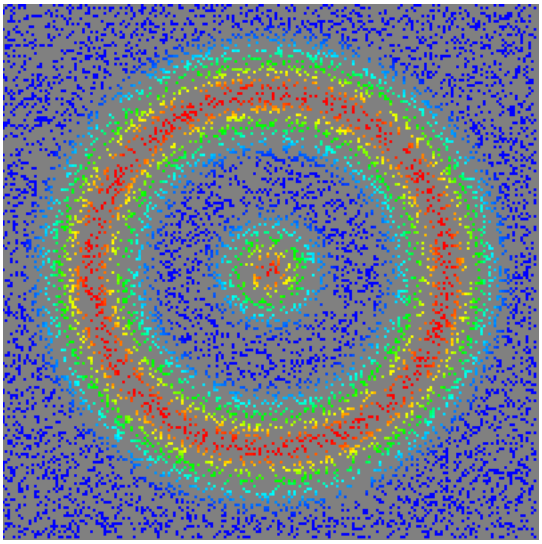


Figure 2.1.1-1: $10{,}000$ samples of the demonstration function by PLAIN

2.1.2 Parameters

PLAIN has no control parameters.

2.1.3 Parallelization

The samples (from the domain) can be drawn in parallel. If they were, then, where p is the number of available processors, the number of rounds of sampling would be $\lceil n/p \rceil$.

2.2 VEGAS

In importance sampling, if $g = \prod_{i=1}^{d} g_i$ (that is, g were separable), then:

$$\int_D g(\mathbf{x})\,d\mathbf{x} = \int_{l_1}^{u_1}\int_{l_2}^{u_2}\cdots\int_{l_d}^{u_d}\prod_{i=1}^{d} g_i(x_1, x_2, \cdots, x_d)\,dx_1\,dx_2\,\cdots\,dx_d$$

$$= \prod_{i=1}^{d}\int_{l_i}^{u_i} g_i(x_i)\,dx_i$$

($x_{j\neq i}$ is constant with respect to dx_i.) VEGAS ([11-12]) does not use such a g

to weight samples because the shape of such a g usually cannot approximate the shape of $|f|$ well (because $|f|$ usually is inseparable). However, VEGAS does use a proxy for such a g for adaptive importance sampling (possibly with stratified sampling).

VEGAS uses d separate computational grids (one grid per dimension) as a proxy for a separable g. Each grid is associated with a different coordinate axis and covers D with bins (that is, cells) that are perpendicular to the axis. Each coordinate of a sample is randomly chosen by, firstly, choosing a random bin along the relevant axis and, secondly, choosing a random offset into the bin. $n = 2{,}000$ samples of the demonstration function and the (overlaid in black) grids with $\mathrm{BINS\_MAX}$ (which equals 50, by default) bins per grid look like:

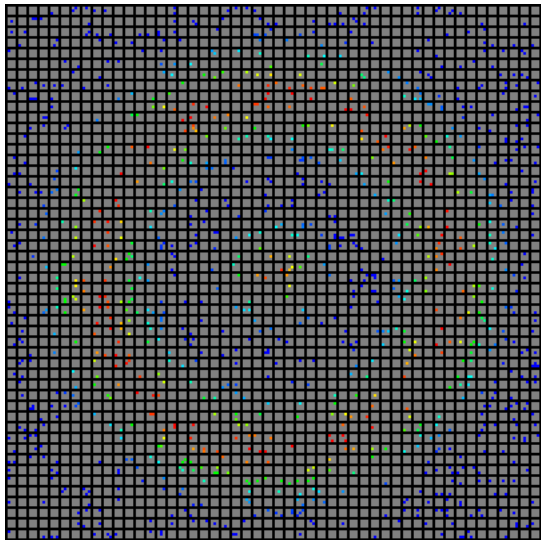


Figure 2.2-1: $n = 2{,}000$ samples of the demonstration function and the grids

Initially, the widths of the bins are equal. After every round of n samples, the widths are varied to adapt the bin density towards $|f|$ using the most recent samples and the following process. Firstly, each bin is associated with a weight that is the sum of the squares of the one-point approximate integrals over the bin. After the first round, the weights of the bins (visualized like a histogram with an arbitrary vertical scale) for each grid look like:

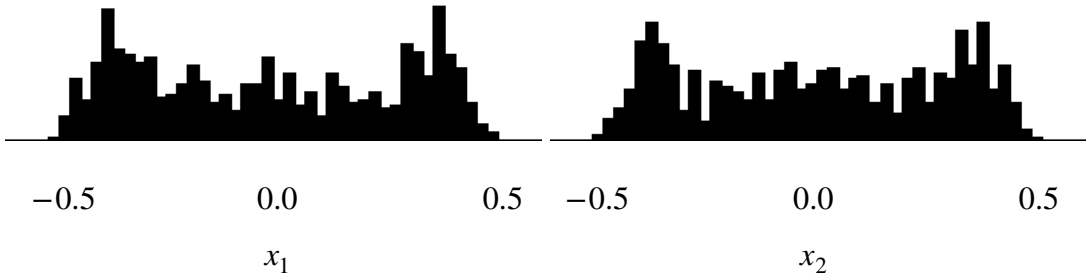


Figure 2.2-2: The weights of the bins for each grid after the first round

The weights (which correspond to the tops of the bars in the visualizations) for each grid are the image of a bin-wise constant function over the grid. The partial integrals over the bins (which correspond to the areas of the bars) are the weights multiplied by the corresponding widths. Lastly, the widths are varied so that the partial integrals of each function are more (depending on the value of the parameter called alpha) equal to the other partial integrals of the function. Collectively, a round and the adaptation process using the samples from the round are an iteration. The second round and the once-adapted grids look like:

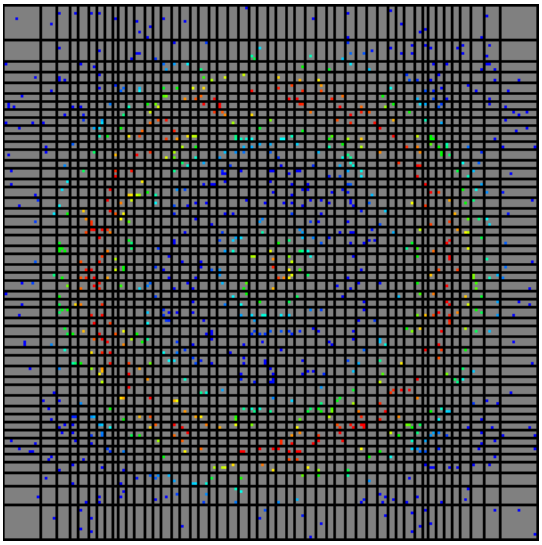


Figure 2.2-3: The second round and the once-adapted grids

After the first iteration, the widths are usually unequal. The adapted widths are narrow/wide where $|f|$ is large/small. After the second round, the weights for each grid look like:

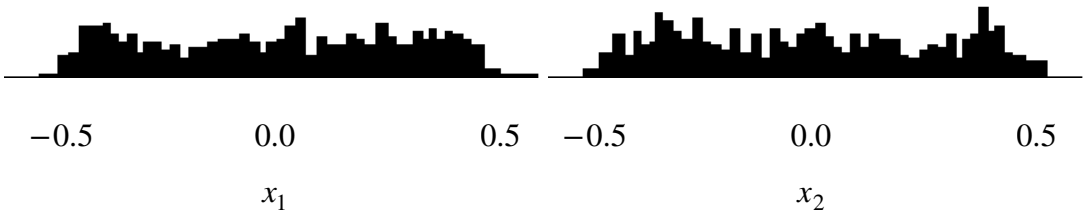


Figure 2.2-4: The weights for each grid after the second round

After the first iteration, the partial integrals in the current grids are usually less disparate than the partial integrals in the previous grids so the process yields diminishing returns. The fifth round and the four-times-adapted grids look like:

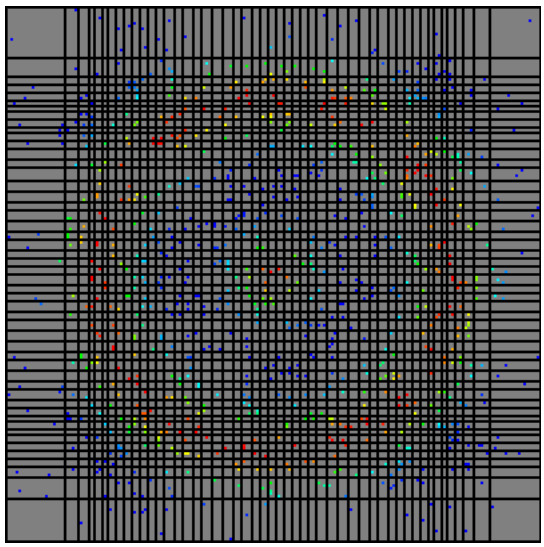


Figure 2.2-5: The fifth round and the four-times-adapted grids

In modes other than importance-sampling-only mode, VEGAS samples boxes (that is, regions defined by the intersections of the bins from each grid) rather than D, which can be cast as a box (which groups all of the intersections). Each box is sampled the same number of times as every other box and sampled at least twice (so that an approximate error in the box can be computed), which may necessitate modification of n. Before the

potential modification of n, if $2\lfloor\sqrt[d]{n/2}\rfloor < \mathrm{BINS\_MAX}$, then VEGAS would use $\lfloor\sqrt[d]{n/2}\rfloor$ boxes per dimension and continue in or fall back (from stratified-sampling mode) to importance-sampling (in boxes) mode. Else, VEGAS uses at least one box per bin, which may necessitate modification of the number of bins and number of boxes, and continues in or switches (from importance-sampling mode) to stratified-sampling mode. In importance-sampling mode, $n = 968$ (modified from $1{,}000$) samples and the grids with $\mathrm{BINS\_MAX}$ bins per grid in 22 boxes (overlaid in light gray) per dimension look like:

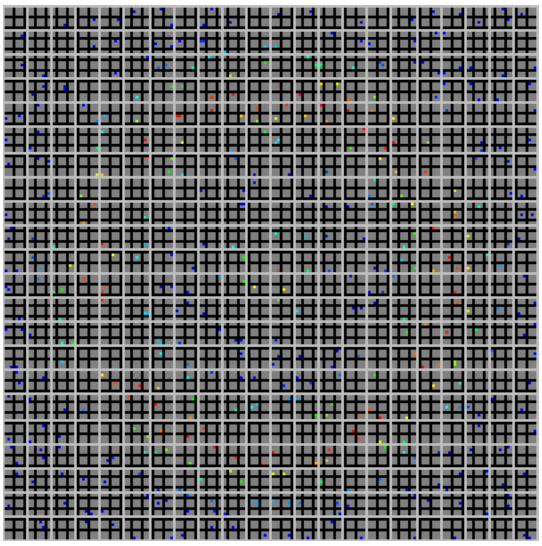


Figure 2.2-6: In importance-sampling mode, $n = 968$ samples and the grids

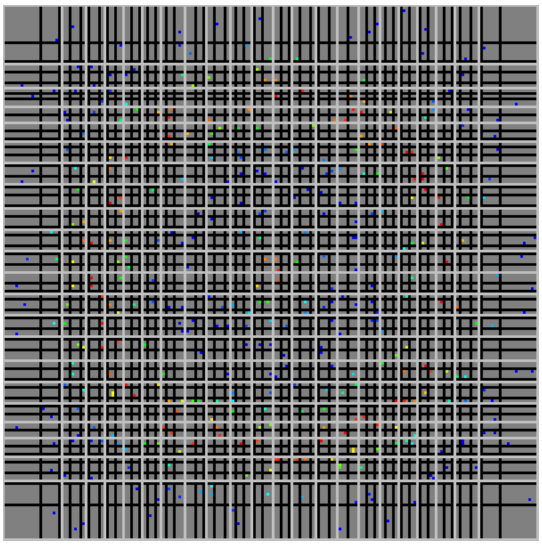


Figure 2.2-7: The second round and the once-adapted grids

After each round, the weights of the boxed bins are similar to the weights of the unboxed bins (in importance-sampling-only mode). The second round and the once-adapted grids (and boxes) look like those in Figure 2.2-7 (above). As in importance-sampling-only mode, the adaptation process yields diminishing returns in importance-sampling mode. In stratified-sampling mode, the process differs from the process in the other modes in the way that each bin is associated with a weight. A weight is the variance of the one-point approximate integrals over the bin (rather than the sum of the squares of the one-point approximate integrals over the bin). In stratified-sampling mode, $n = 1922$ (modified from $2{,}000$) samples and the grids with 31 bins (modified from $\mathrm{BINS\_MAX}$) per grid in 31 boxes per dimension look like:

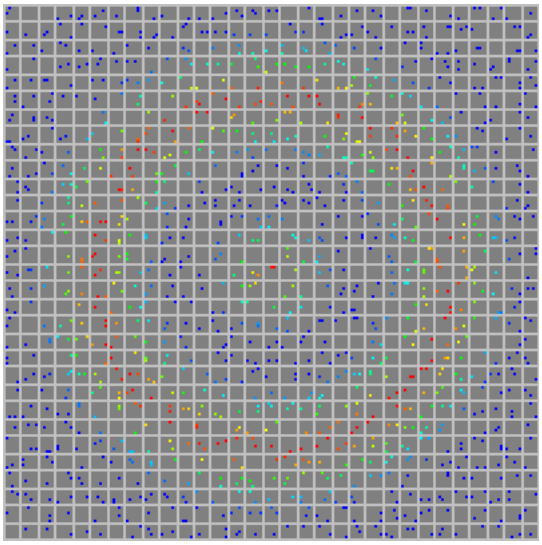


Figure 2.2-8: In stratified-sampling mode, $n = 1922$ samples and the grids

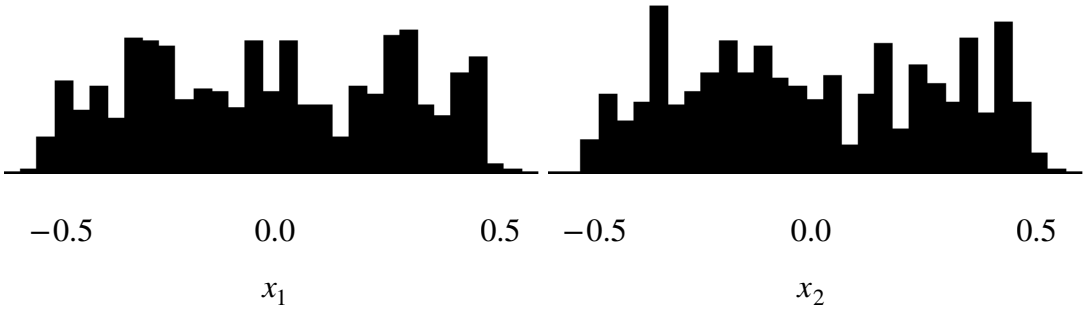


Figure 2.2-9: The weights for each grid after the first round

After the first round, the weights (with a new arbitrary vertical scale) for each grid look like those in Figure 2.2-9 (above). The second round and the once-adapted grids (and boxes) look like:

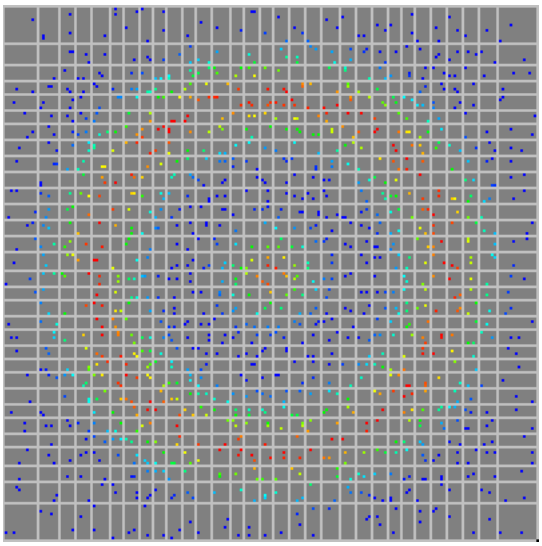


Figure 2.2-10: The second round and the once-adapted grids

Clearly, the boxes are tied to the grids. After the second round, the weights for each grid look like:

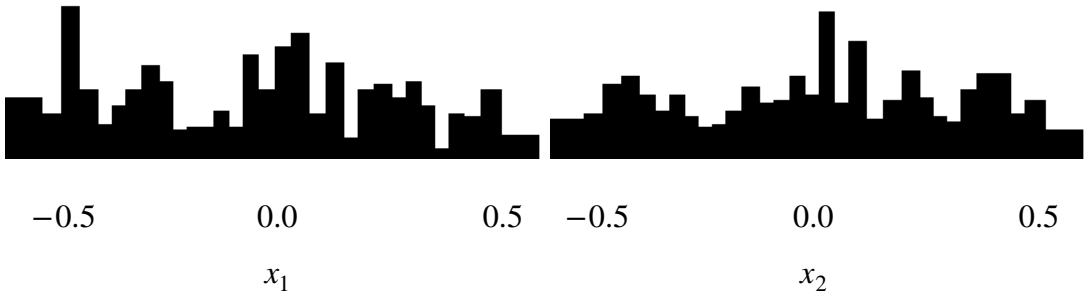


Figure 2.2-11: The weights for each grid after the second round

For each iteration, where B is the number of bins to the dth power (that is, the number of areas defined by the intersections of the bins from each grid) and $A_{\text{bins}(\mathbf{x})}$ is the area defined by the intersection of the bins containing $\mathbf{x}$:

$$w_i = \frac{B}{n} A_{\text{bins}(\mathbf{x}_i)}$$

That is, w_i equals $A_{\text{bins}(\mathbf{x}_i)}$ scaled so that $\sum_{i=1}^{n} w_i \simeq A_D$. For error computation, where m is the number of samples per box, the approximate integral over each box is:

$$\tilde{I}_{\text{box}} = \frac{\sum_{\mathbf{x}\in\text{box}} \frac{B}{n} A_{\text{bins}}(\mathbf{x}) f(\mathbf{x})}{m}$$

$\tilde{I}_{\text{box}}$ is the mean of m one-point approximate integrals. The standard error in $\tilde{I}_{\text{box}}$ is:

$$E_{\text{box}} = \sqrt{\frac{\sum_{\mathbf{x}\in\text{box}} (\frac{B}{n} A_{\text{bins}(\mathbf{x})} f(\mathbf{x}) - \tilde{I}_{\text{box}})^2}{m(m-1)}}$$

The total approximate error over the boxes used by VEGAS is:

$$\tilde{E} = \sqrt{\sum_{\text{boxes}} E_{\text{box}}^2}$$

Note that, if there were unequal numbers of bins and boxes per dimension, then $A_{\text{bins}(\mathbf{x})}$ would not equal the area of the box containing $\mathbf{x}$, which should be used. VEGAS returns $\tilde{I}$ and $\tilde{E}$ accumulated over k iterations and χ^2/k for $\tilde{I}$ (that is the χ^2 value for the k values of $\tilde{I}$ per degree of freedom).

2.2.1 Demonstration

$10{,}000$ (that is, 5 iterations of $2{,}000$) samples of the demonstration function by VEGAS in importance-sampling-only mode look like those in Figure 2.2.1-1 (below). Note that both features are well resolved but the unimportant regions inside the (imaginary) square circumscribed about the outer feature are sampled as frequently as the important regions because of the limitations of the adaptation process. Note well that the process cannot avoid sampling unimportant regions bounded by (curving) features, such as the valley between the features.

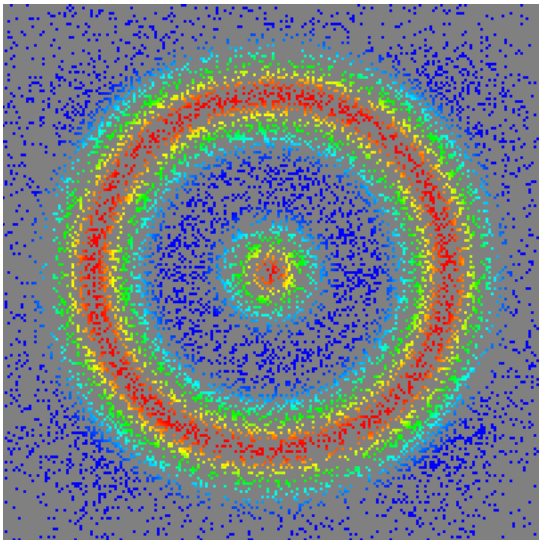


Figure 2.2.1-1: $10{,}000$ samples of the demonstration function by VEGAS in importance-sampling-only mode

$9{,}610$ (that is, 5 iterations of $1{,}922$) samples of the demonstration function by VEGAS in stratified-sampling mode looks like:

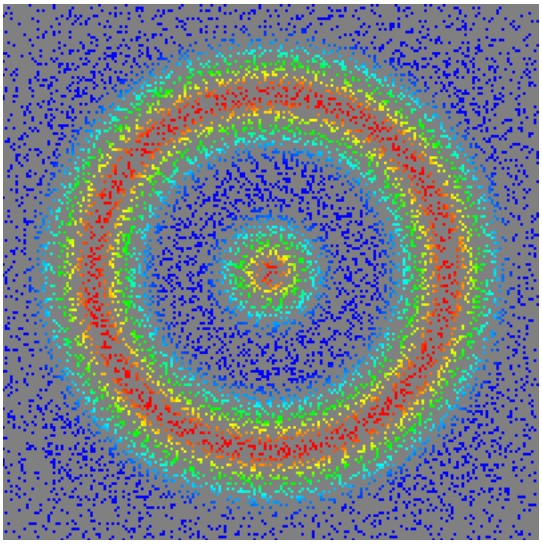


Figure 2.2.1-2: $9{,}610$ samples of the demonstration function by VEGAS in stratified-sampling mode

Note that the samples are more smoothly distributed than the samples in importance-sampling-only mode but suffer from the limitations of the adaptation process too.

Because VEGAS samples according to a separable distribution, it can be

inefficient.

2.2.2 Parameters

VEGAS has the following control parameters:

Name	Description	Default Value
alpha	It controls the stiffness of the adaptation algorithm and typically has a value between 1 and 2. If it were to have a value of 0, then the grids would not be adapted.	1.5
iterations	It controls the number of iterations.	5
mode	It controls the sampling mode. The possible values of it are GSL_VEGAS_MODE_IMPORTANCE_ONLY, GSL_VEGAS_MODE_IMPORTANCE, GSL_VEGAS_MODE_STRATIFIED.	GSL_VEGAS_MODE_IMPORTANCE
stage	It controls the stage of the method. If it were to have a value of: • 0, then the method would begin anew. • 1, then the method would use the grids from the previous run of the method but discard the accumulated results (which would enable preadaptation of the grids). • 2, then the method would use the accumulated results and grids with the same bin density as the grids from the previous run. • 3, then the method would use everything from the previous run (which would enable addition of iterations).	0

Table 2.2.2-1: VEGAS' control parameters

In the rest of this thesis, the default values are used as would be done by a novice user, although expert use of the stage parameter could improve results.

2.2.3 Parallelization

During each iteration, the samples from the current grid can be drawn in parallel. If they were, then, where p is the number of available processors and $g = \text{iterations}$ is the number of grids, the number of rounds of sampling would be $g\lceil n/p \rceil$. Although the total number of samples is gn (rather than n), typically $n \gg g$. Therefore, the parallelism typically is high.

2.3 MISER

MISER ([13]) does recursive stratified sampling on a *budget* of n evaluations that are allocated among the new (hyper-)rectangular subdomains (that is, strata) of D that are created at each recursive step according to estimates of the variance of f in each new subdomain. If a (sub)domain were allocated at least the minimum number of evaluations required for stratification (which is the value of the parameter called min_calls_per_bisection), then the (sub)domain would be decomposed (that is, stratified) into two subdomains by cutting the (sub)domain perpendicularly with respect to one of its dimensions such that the estimate of the total variance in the subdomains is minimal given the cut candidate for each dimension. Else, $\tilde{I}$ and $\tilde{E}$ are computed for the (sub)domain by way of PLAIN. Finally (for D):

$$\tilde{I} = \sum_{\text{subdomains}} \tilde{I}_{\text{subdomain}}$$

$$\tilde{E} = \sqrt{\sum_{\text{subdomains}} \tilde{E}^2_{\text{subdomain}}}$$

Consider the following diagram of a (sub)domain being cut:

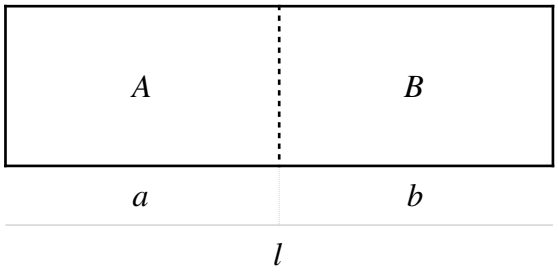


Figure 2.3-1: A (sub)domain being cut

A is the subdomain on the low side of the selected cut, B is the subdomain on the high side of the selected cut, a is the length of A in the cut dimension, b is the length of B in the cut dimension, and l is the length of the (sub)domain in the cut dimension (therefore, $l = a + b$). Furthermore, m is

the minimum number of evaluations required for the estimation of the variance in a subdomain (which is the value of the parameter called min_calls), n^* is the remaining number of evaluations, and α (which corresponds to the parameter called alpha) is positive. The allocation is:

$$n_A = m + (n^* - 2m)\frac{\frac{a}{l}\sigma_A^{2/(1+\alpha)}}{\frac{a}{l}\sigma_A^{2/(1+\alpha)} + \frac{b}{l}\sigma_B^{2/(1+\alpha)}}$$

$$n_B = m + (n^* - 2m)\frac{\frac{b}{l}\sigma_B^{2/(1+\alpha)}}{\frac{a}{l}\sigma_A^{2/(1+\alpha)} + \frac{b}{l}\sigma_B^{2/(1+\alpha)}}$$

By default, $m = 16d$ (that is, 8 points on both sides of each cut candidate). Based on empirical evidence, the authors of MISER recommend setting the value of α to 2, which is the default value.

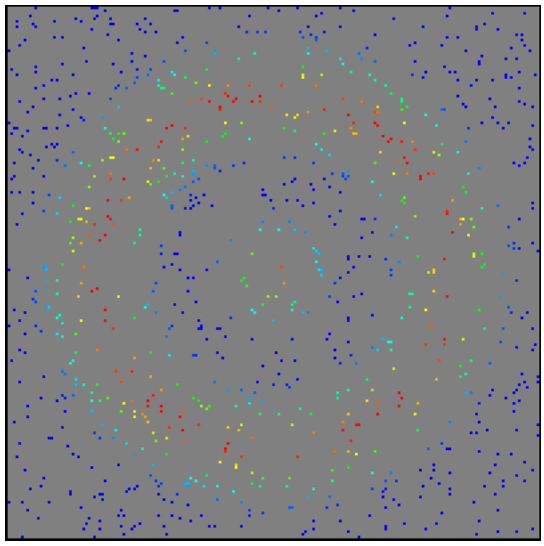


Figure 2.3-2: $0.1n = 1{,}000$ samples and the boundaries of D

If n^* were greater than the value of min_calls_per_bisection, which is $32m$ by default, then the greater of m and a certain fraction (which corresponds to the parameter called estimate_frac) of n^* evaluations would be used to estimate the variances. By default, the fraction is 0.1. Initially, $n^* = n$. On a budget of $n = 10{,}000$ evaluations of the demonstration function (for which

$m = 32 \Rightarrow 32m = 1{,}024 < n^*$), $0.1n = 1{,}000$ evaluations are used to estimate the variances in D. The samples and the boundaries (overlaid in black) of D look like those in Figure 2.3-2 (above). The d default cut candidates are through the center of the (sub)domain to be stratified. If the value of the parameter called dither were set to a nonzero value, then the $2d$ dithered cut candidates would be offset from the center by the length of the (sub)domain in the cut dimension scaled by the value. The cut candidates (overlaid in black) without (which is the default) and with dithering look like:

$\text{dither} = 0$ $\text{dither} = 0.1$

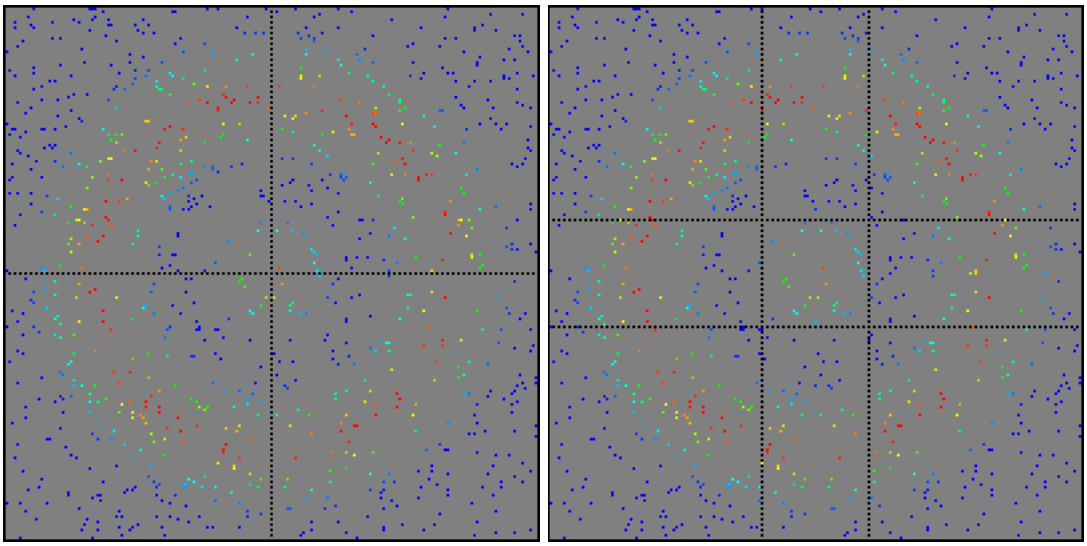


Figure 2.3-3: The cut candidates without and with dithering

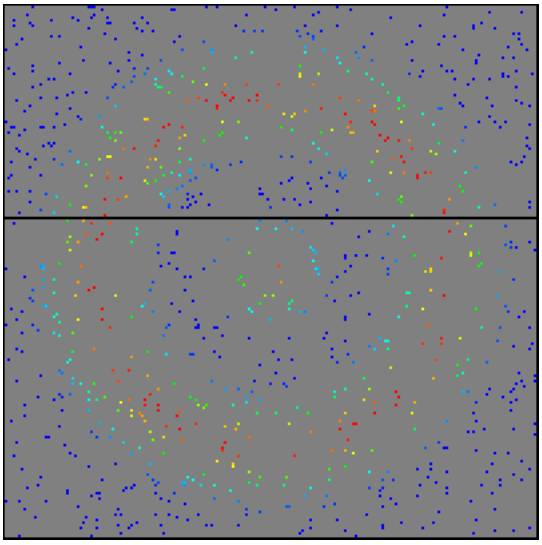


Figure 2.3-4: The samples and the boundaries of the subdomains after a cut candidate is selected

0.1 is a typical nonzero value for dither. Although there are two dithered cut candidates per dimension, only one is randomly selected and used per stratification. Dithering enables MISER to break symmetries in f that would hinder the allocation of evaluations and, therefore, should be used with (possibly) symmetric f. Herein, dither is set to 0.1. After a cut candidate is selected, the samples and the boundaries of the subdomains look like those in Figure 2.3-4 (above). Of the remaining $9{,}000$ evaluations, $5{,}773$ evaluations are allocated to the lower subdomain and $3{,}226$ evaluations are allocated to the upper subdomain. Note that the allocation process is lossy (that is, $5{,}773 + 3{,}226 \neq 9{,}000$) so fewer than n evaluations may be spent. Subsequently, the lower subdomain is recursively stratified twice and the upper subdomain is recursively stratified six times. After the $9{,}991$st and last evaluation, the samples and the boundaries of the subdomains look like:

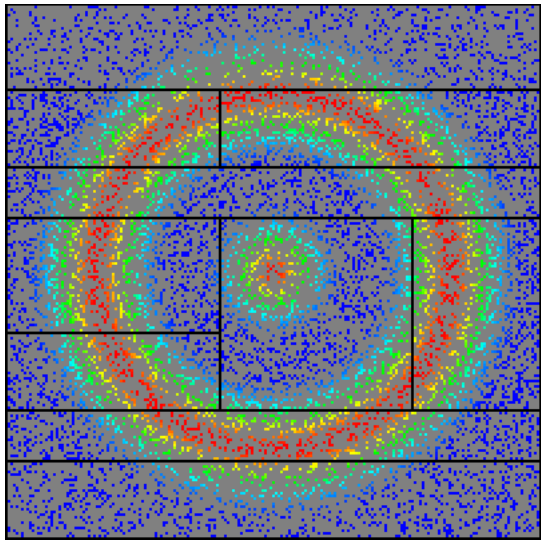


Figure 2.3-5: The samples and the boundaries of the subdomains after the $9{,}991$st and last evaluation

There are only 10 subdomains. The number of subdomains could be increased by decreasing the value of estimate_frac, min_calls, or min_calls_per_bisection. However, doing so may also decrease the accuracy of the results. On a budget of $n = 100{,}000$ evaluations of the demonstration function, MISER spends $99{,}915$ evaluations and creates 86 subdomains. The samples corresponding to the evaluations and the

boundaries of the subdomains look like:

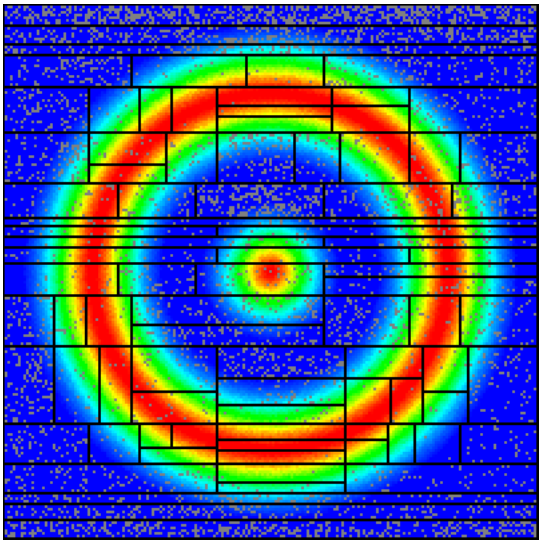


Figure 2.3-6: The samples corresponding to $n = 100{,}000$ evaluations and the boundaries of the subdomains

MISER is capable of stratifying a degeneracy but perhaps only on an onerous evaluation budget. Moreover, an evaluation budget is inappropriate for the applications that are targeted by this thesis in which accuracy (rather than duration) is paramount. For the applications, duration should be unlimited but short on account of efficient (rather than limited) evaluation.

2.3.1 Demonstration

$9{,}991$ samples of the demonstration function by MISER look like:

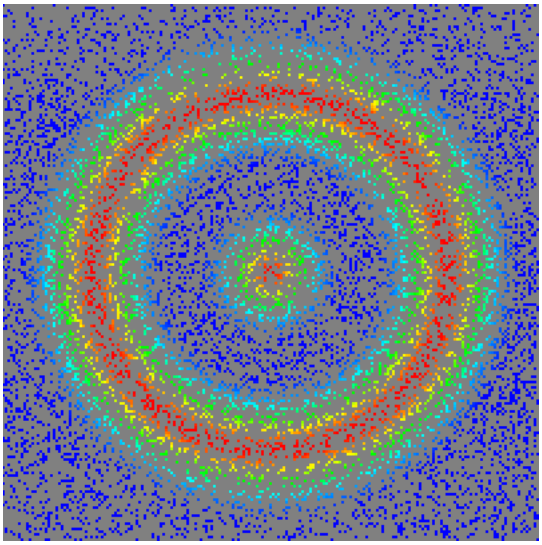


Figure 2.3.1-1: $9{,}991$ samples of the demonstration function by MISER

Note that both features are well resolved but the unimportant regions excluding bands above and below the outer feature are sampled as frequently as the important regions because of the coarseness of the strata. Because MISER samples in potentially coarse strata, it can be inefficient.

2.3.2 Parameters

MISER has the following control parameters:

Name	Description	Default Value
alpha	It controls the allocation of evaluations among sibling strata and must be positive.	2
dither	It controls, for each dimension, a randomly signed deviation of the potential position of stratification from 0.5 of the length of the stratum to be stratified (which enables symmetry breaking). It must be within 0.5 of 0. A typical nonzero value of dither is 0.1.	0
estimate_frac	It controls the fraction of the number of currently available evaluations that are used per estimate of the variance. It must be positive.	0.1
min_calls	It controls the minimum number of evaluations that are used per estimate of the variance. It must be positive.	$16d$
min_calls_per_bisection	It controls the minimum number of evaluations that are used per stratification. It must be positive.	32min_calls

Table 2.3.2-1: MISER's control parameters

In the rest of this thesis, the default values are used except 0.1 for the value of dither because f is symmetric.

2.3.3 Parallelization

During each recursive step, the samples from the current stratum can be drawn in parallel. If they were, then, where p is the number of available processors and $s = \lfloor n/\mathrm{min\_calls\_per\_bisection} \rfloor$ is the maximum number of strata, the maximum number of rounds of sampling would be $s\lceil \mathrm{min\_calls\_per\_bisection}/p \rceil$. Using the default value of $\mathrm{min\_calls\_per\_bisection}$ yields plenty of parallelism. However, a user may use a lesser value to increase s for a given n.

3 Nested Sampling

The original implementation of nested sampling is Nest. The most frequently used (Monte Carlo) method for (multidimensional) numerical integration in cosmological model selection is MultiNest, which is based on nested sampling.

Nested sampling ([14-20]) is an algorithm to compute $\tilde{I}$ while maximizing smooth, positive f (on D) by successively sampling the region(s) bounded by sets of successively higher-valued points. Such sets that do not share a boundary with D are nested within the sets that only contain lower-valued points (because f is smooth). The level sets of the demonstration function that only contain the points with the values of 0.5 and 1.0 (which correspond to green and red) look like:

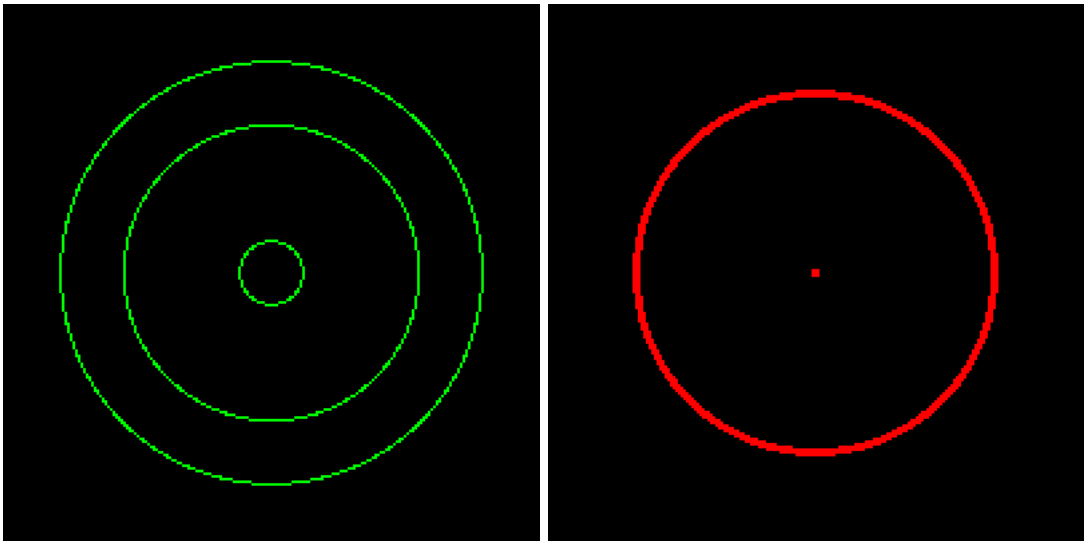


Figure 3-1: The level sets of the demonstration function that only contain the points with the values of 0.5 (at left) and 1.0 (at right)

Note that the sets are not necessarily connected and the set of higher-valued (red) points nests within the set of lower-valued (green) points. If either the circular or the outer annular region that is bounded by the lower-valued points were sampled, then the resulting sample would be nearer than the points to the higher-valued points, which are the highest-valued points of the

function and, therefore, the objectives of the maximization process. If a (nonempty) set of points were bounded by the level set that contains the sample, then the set would be subsequently sampled.

If a level set of f were infinitesimally thin, then the integral of f over the set would be zero (regardless of the value of the points in the set). However, the integral over the set of points in D between two disjoint level sets (which is composed of infinitely many level sets) is nonzero (because f is positive). The subdomains (that is, disjoint sets of points) of the demonstration function that only contain the points with values between 0.0 (which corresponds to blue) and 0.5 and between 0.5 and 1.0 look like:

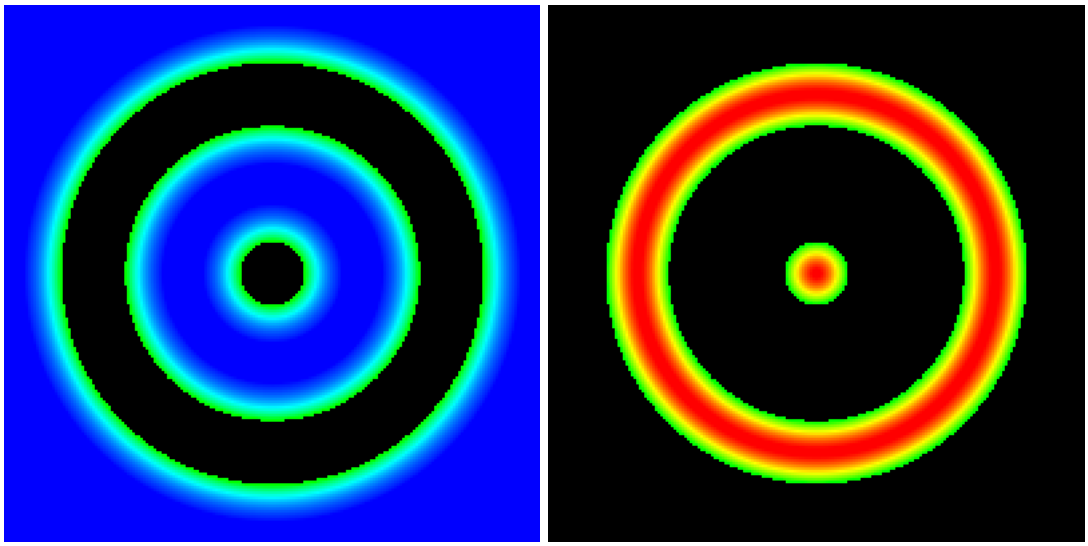


Figure 3-2: The subdomains of the demonstration function that only contain the points with values between 0.0 and 0.5 (at left) and between 0.5 and 1.0 (at right)

Clearly, the sum of the integrals over the subdomains equals the integral over the domain of the demonstration function. If only the level sets at 0.5 and 1.0 were known, then the integral over the domain could be crudely approximated by the sum of 0.5 times the area that is not bounded by the level set at 0.5, and 1.0 times the area that is not bounded by the level set at 1.0 but is bounded by the level set at 0.5. If more (distinct) level sets were known, then the integral could be less crudely approximated. For nested level sets of f that, where i is the degree of nesting, contain $\mathbf{x}_i$ and bound an

area of A_i, where $A_0 = A_D$:

$$w_i = A_{i-1} - A_i$$

Because, in practice, A_i cannot practically be measured, the nested sampling algorithm assumes that, where $N \in Z^+$ is a parameter that controls the number of samples that are uniformly distributed in A_i, $A_i \cong A_D e^{-i/N}$ (that is, A_i is a fraction of A_D that, as i increases, exponentially decays at a rate controlled by N). Thus:

$$w_i \cong \left(e^{-\frac{i-1}{N}} - e^{-\frac{i}{N}}\right) A_D$$

That is, w_i is a fraction of A_D that, as i increases, decreases at a rate controlled by N. The dependence of w_i on N is demonstrated by:

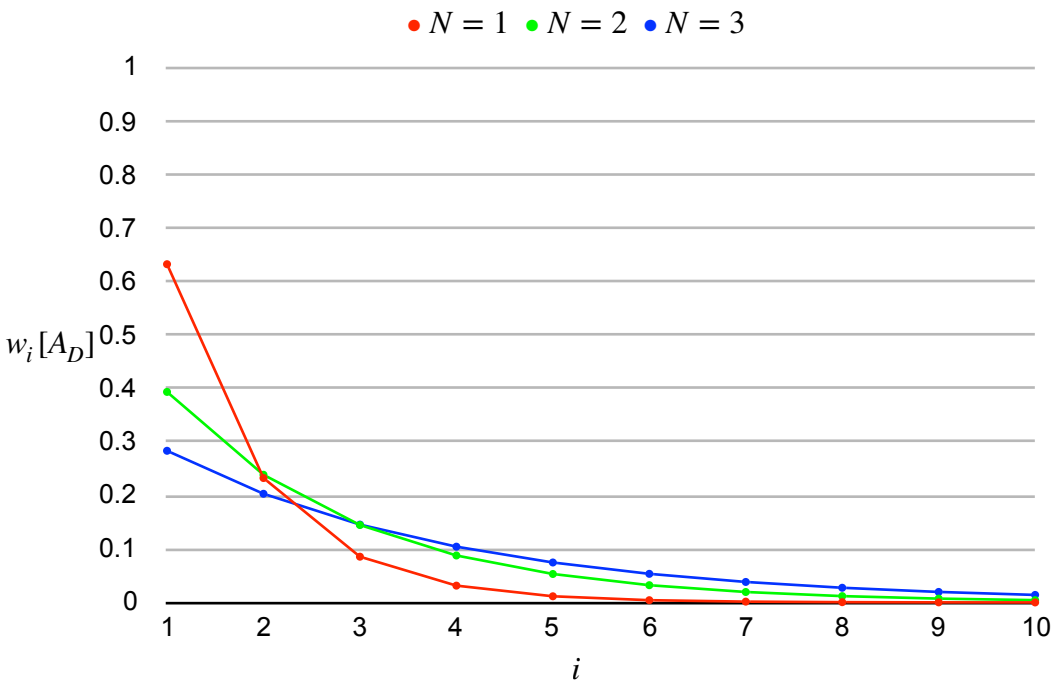


Figure 3-3: The dependence of w_i on N

As N is incremented, w_i versus i flattens. The flatness of w_i versus i interestingly affects the accumulated weight, $W_i = \sum_{j=1}^{i} w_j$. The (indirect) dependence of W_i on N is demonstrated by:

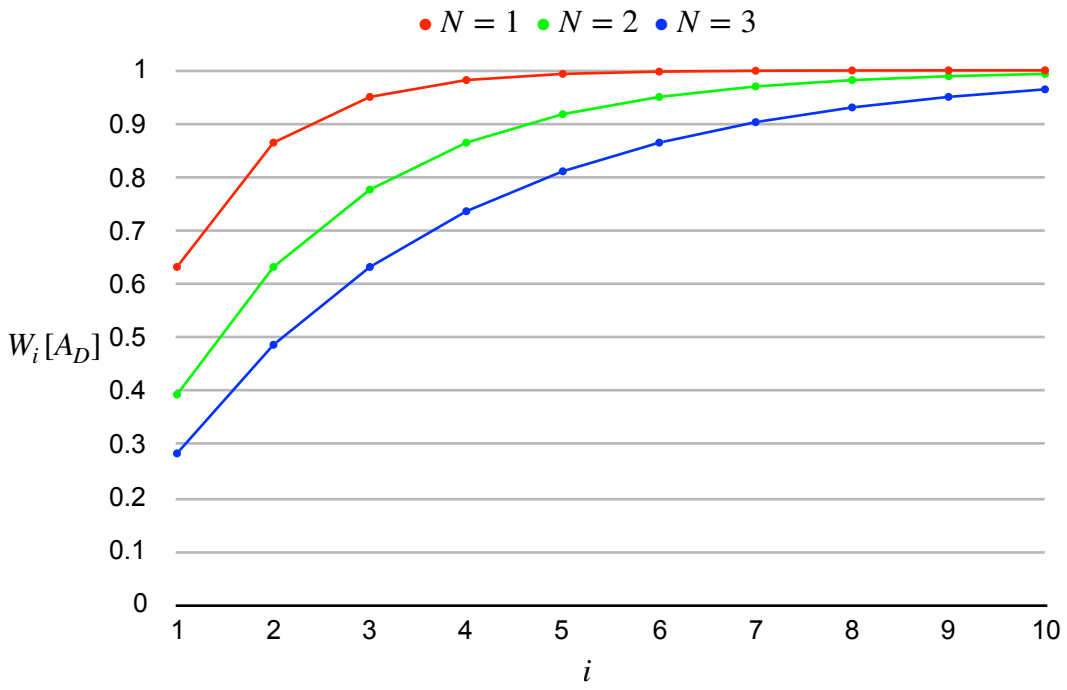


Figure 3-4: The dependence of W_i on N

As N is incremented, W_i shrinks. The shrinking of W_i reflects the redistribution of some of the total weight from less-nested samples to more-nested samples. As $i \to \infty$, $w_i \to 0.0$ and, therefore, $W_i \to A_D$. That is, $W_i \sim A_D$. However, to double precision, $w_i = 0.0$ for $i \geq i^*$. The linear fit through i^* for $N = 1..3$ is:

$$i^* \cong 739.74N + 36.928 \, (\mathrm{R}^2 = 1)$$

Effectively, $\tilde{I}$ is limited to i^* nonzero terms. N should be such that i^* is large enough to enable $\tilde{I}$ to be accurate. n should not be greater than i^*. If $W_n \ll A_D$, $\tilde{I}$ should be corrected (because $\tilde{I}$ is incomplete) thus, where v is an appropriate value:

$$\tilde{I} \leftarrow \tilde{I} + (A_D - W_n)v$$

Described in the rest of this chapter are the implementations of nested sampling Nest (in Section 3.1), which is the original implementation, and MultiNest (in Section 3.2), which is related to but has a more efficient approach to the sampling process than Nest.

3.1 Nest

In Nest ([21]), there is the set of the samples, S, and the set of the top N (which corresponds to the parameter called Nobjects) highest-valued samples, T, where $T \subseteq S$. The sampling process begins by sampling D (according to some distribution function) N times. $N = 10$ uniformly distributed samples of the demonstration function, where the lowest-valued sample is circled, look like:

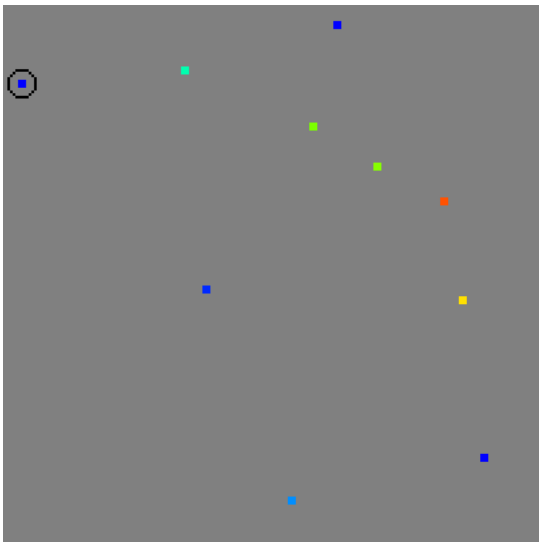


Figure 3.1-1: $N = 10$ uniformly distributed samples of the demonstration function, where the lowest-valued sample is circled

The samples are the elements of T and of S (that is, $T = S$). The sampling process continues by sampling the region(s) bounded by the level set that contains the lowest-valued sample in T. Because, in practice, the level sets are unknown, Nest must resort to a sampling technique that does not require knowledge of the level sets. Such a technique is rejection sampling with respect to the target value, in which the function is uniformly sampled until a sample with a higher value than the target value is found (that is, accepted while the other samples are rejected). After a new (higher-valued) sample is accepted and replaces the lowest-valued sample in T, the samples in T and the samples in S, where the new sample is circled in white and the current lowest-valued sample in T is also circled, look like:

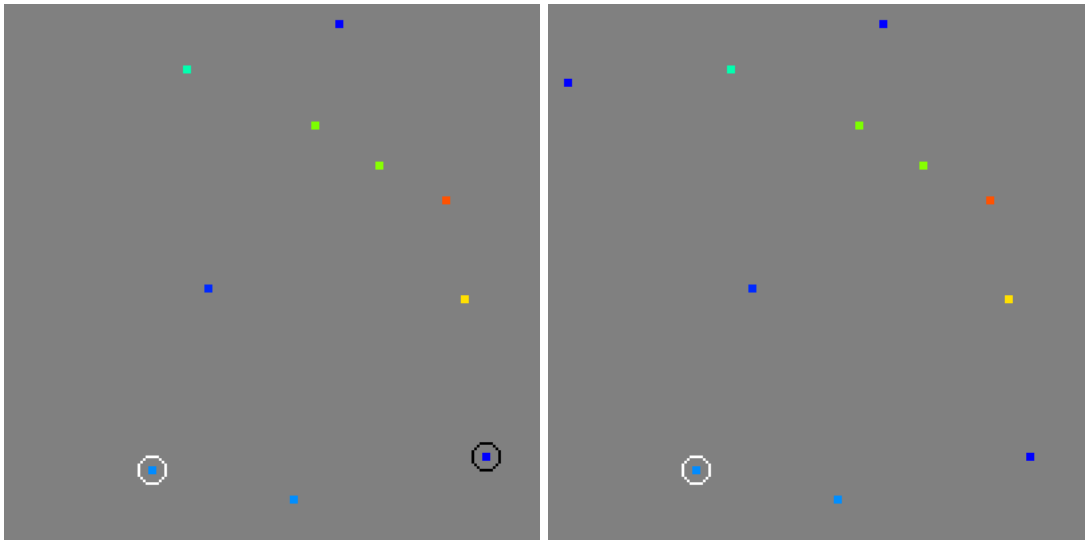


Figure 3.1-2: The samples in T (at left) and the samples in S (at right), where the new sample is circled in white and the current lowest-valued sample in T is also circled, after a new sample is accepted and replaces the lowest-valued sample in T

The sample that was replaced is not in T but is in S (that is, $T \subset S$). After a (lower-valued) sample is rejected (and does not replace the lowest-valued sample in T), T and S look like:

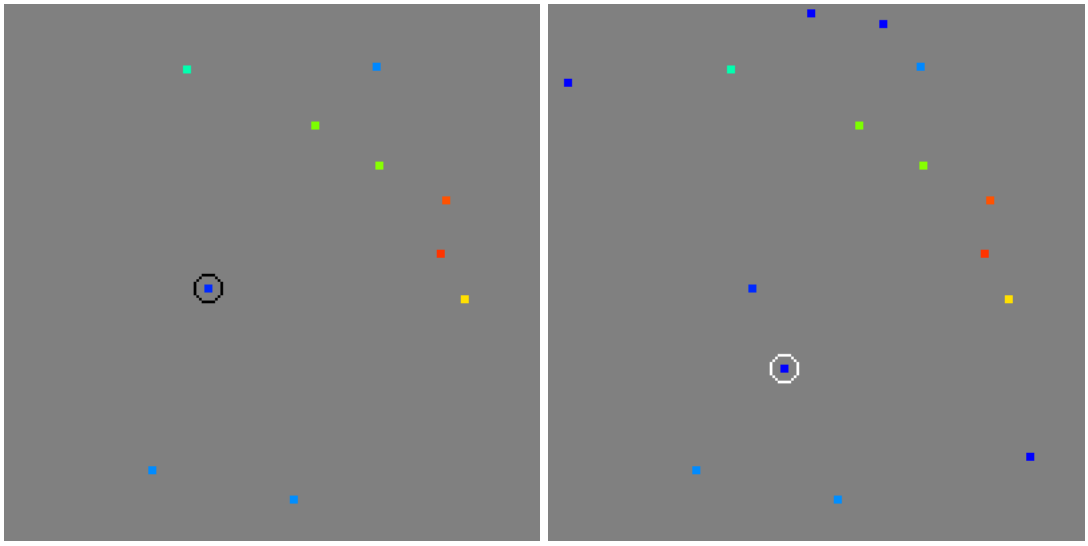


Figure 3.1-3: T (at left) and S (at right) after a sample is rejected

The new sample is not in T but is in S. The acceptance ratio of the sampling process, which has been 1.0, is $14/15 = 0.9\overline{3}$. As the sampling process continues, the ratio typically decreases because finding a sample with a higher value than the current lowest-valued sample in T becomes

increasingly unlikely. After the 80th and last (for $n = 10{,}000$ samples) sample is accepted, $8{,}651$ samples (for which the ratio is less than $1\ \%$) and T and S look like:

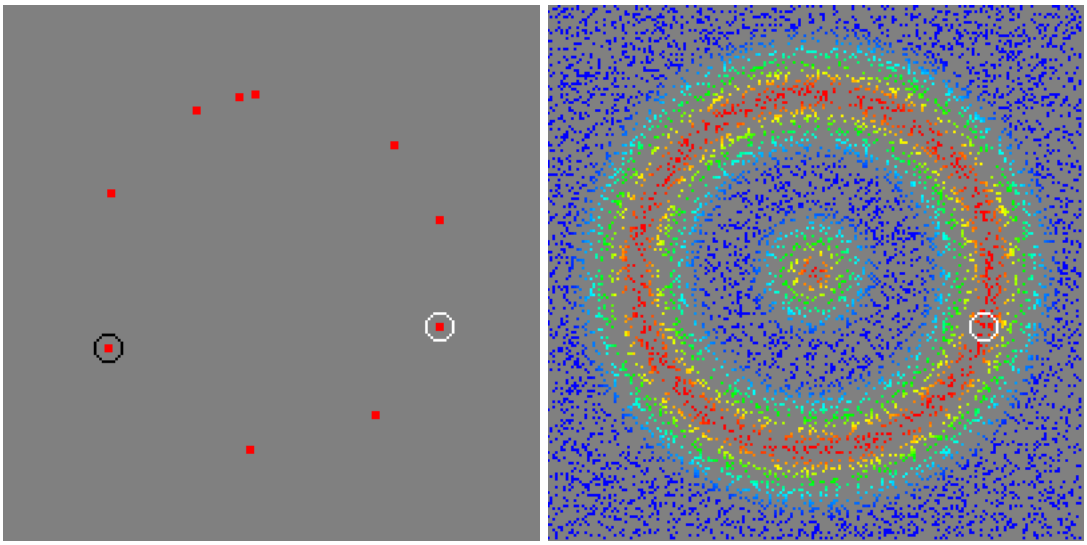


Figure 3.1-4: $8{,}651$ samples and T (at left) and S (at right) after the 80th and last sample is accepted

The samples in T are near the top of the outer feature. The value of the lowest-valued sample in T is approximately 0.999991, which is extremely high. (The value of the highest-valued sample is less than 1.0.) The 70 replaced samples look like:

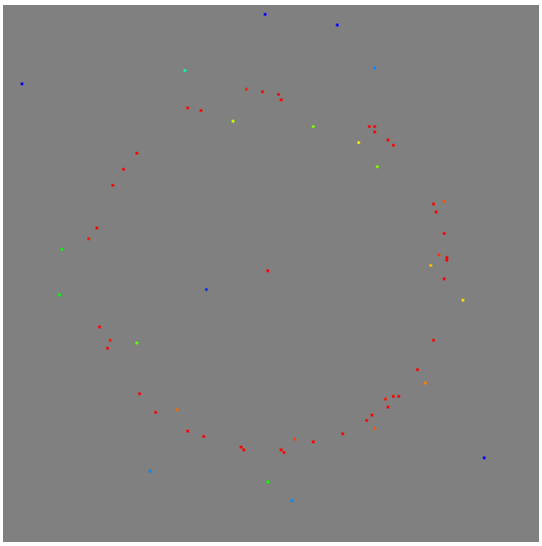


Figure 3.1-5: The 70 replaced samples

The samples are indexed (by i) in replacement order and used to compute $\tilde{I}$

(with weights that are less crude and simple than w_i). The average value of the samples in T is used as v (to correct $\tilde{I}$).

Finally:

$$\tilde{E} = \frac{\sum_{i=1}^{n} w_i}{n} \max_{i=1}^{n} (f(\mathbf{x}_i))$$

That is, $\tilde{E}$ is the average weight times the maximum value, which is an upper bound on $\tilde{I}$.

3.1.1 Demonstration

$10{,}000$ samples of the demonstration function by Nest look like:

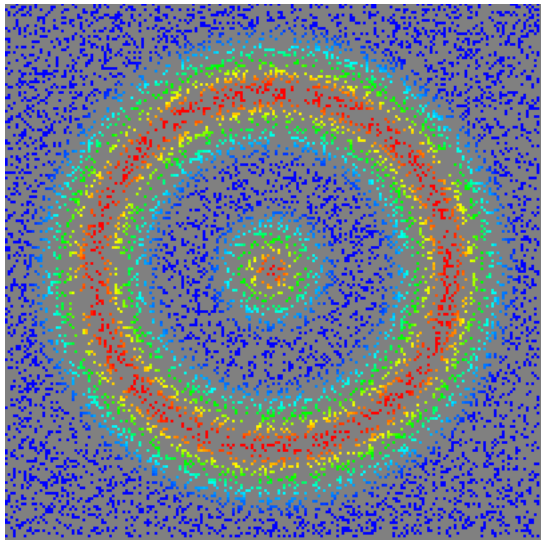


Figure 3.1.1-1: $10{,}000$ samples of the demonstration function by Nest

Note that both features are resolved but the unimportant regions, including the regions near the boundaries of the domain, are sampled as frequently as the important regions. When Nest samples uniformly, it is inefficient.

3.1.2 Parameters

Nest has the following control parameters:

Name	Description
Nobjects	It controls N, the number of objects in T. It must be a positive integer.

Name	Description
MaxSteps	It controls the maximum number of steps (that is, replacements). It must be an integer. It is inactive when negative.
tol	It controls the error tolerance. It must be in [0, 1]. It is inactive when zero.

Table 3.1.2-1: Nest's control parameters

Choosing N such that the accuracy is high and the number of samples is low is challenging. In practice, multiple runs with different values of N may be necessary to balance the accuracy and the number of samples.

3.1.3 Parallelization

During each nested sampling step, where p is the number of available processors, up to p candidate samples from the domain can be drawn in parallel until at least one is acceptable. If they were, then one should be randomly accepted from among the acceptable ones (which would raise the acceptance threshold) until none are acceptable. This could yield a parallel sampling efficiency (that is, the ratio of the number of accepted samples to the number of rounds of sampling) that is much higher than the regular sampling efficiency (that is, the ratio of the number of accepted samples to the number of samples).

Note: If the accepted samples were not randomly chosen from among the acceptable samples, then $\tilde{I}$ could be biased because the sample could be improperly weighted.

3.2 MultiNest

As for Nest, the sampling process of MultiNest ([22]) begins by sampling D N (which corresponds to the parameter called nest_nlive) times. $N = 1{,}000$ uniformly distributed samples of the demonstration function look like those in Figure 3.2-1 (below). As for Nest, the samples are the elements of T and S (which are equal when the number of samples is less than or equal to N).

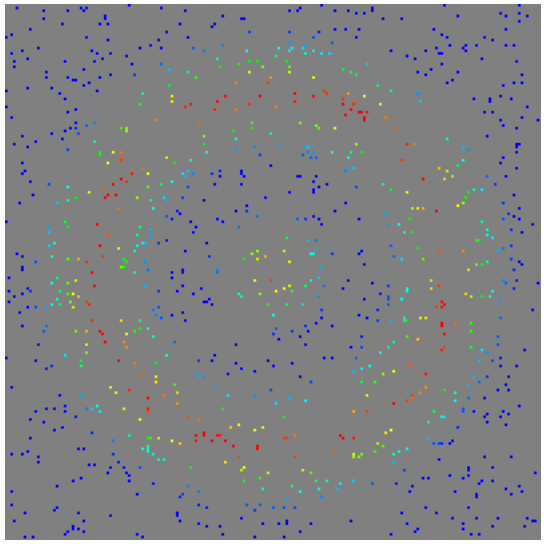


Figure 3.2-1: $N = 1{,}000$ uniformly distributed samples of the demonstration function

The sampling process continues as for Nest until an ellipsoid that contains T is, after enlargement by a factor (which corresponds to the reciprocal of the parameter called nest_efr), contained by D. With a factor of 0.3, such an ellipse exists after $3{,}544$ samples (with $1{,}254$ replacements), when the samples in T and the samples in S, where the ellipse is overlaid (in black), look like:

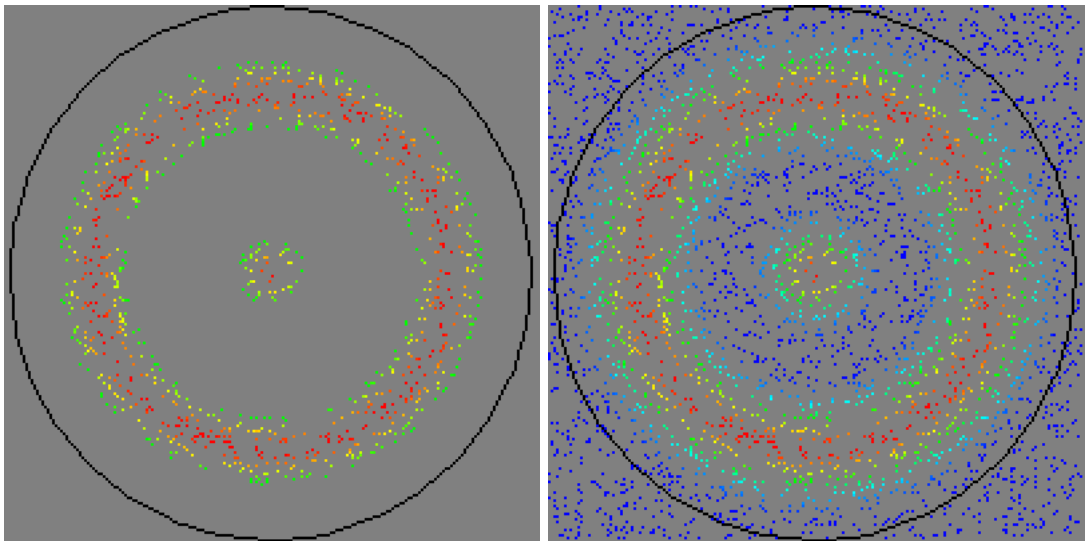


Figure 3.2-2: The samples in T (at left) and the samples in S (at right), where an ellipse is overlaid

Subsequent samples are uniformly drawn from the ellipsoid until a replacement is found. CosmoNest ([23-24]), which inspired MultiNest,

introduced the idea of sampling from an ellipsoid that contains T to increase the acceptance ratio of the sampling process that is described in Section 3.1. The ellipsoid is a proxy for the (unknown) level set that contains the lowest-valued sample in T. To ensure the accuracy of $\tilde{I}$, the level set must be contained by the ellipsoid. Hence, CosmoNest also introduced the enlargement factor. When a replacement is found by MultiNest, the ellipsoid shrinks if the tightest bounding ellipsoid of T shrinks or the enlargement factor is reduced. After $9{,}789$ samples (with $2{,}828$ replacements), T and S with the totally shrunken ellipse look like:

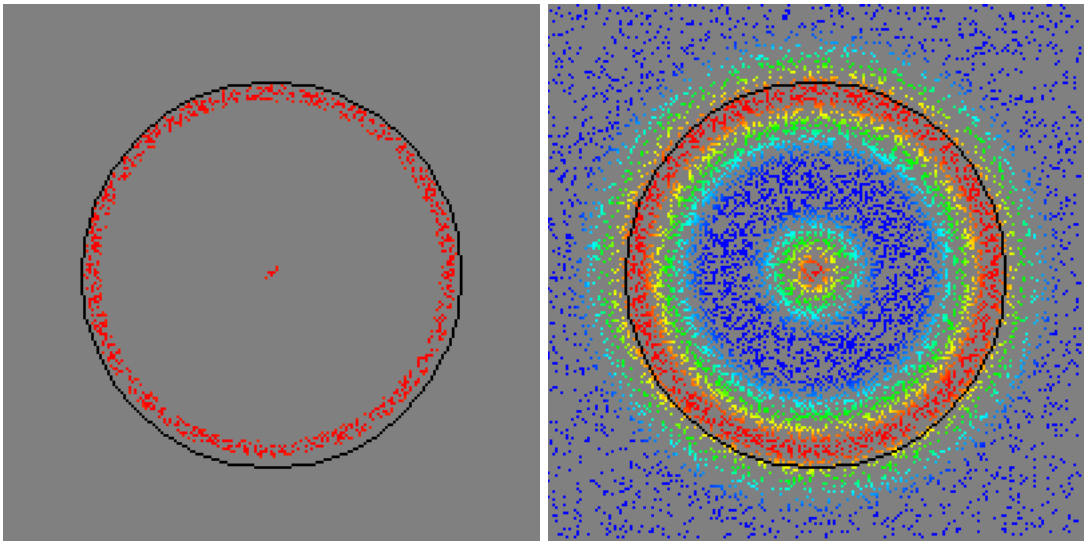


Figure 3.2-3: T (at left) and S (at right) with the totally shrunken ellipse

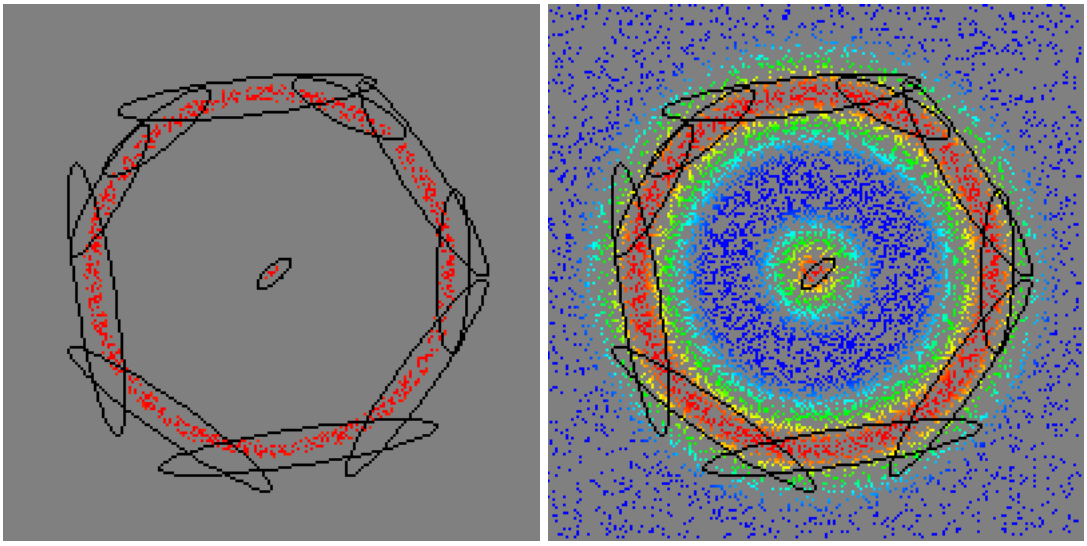


Figure 3.2-4: T (at left) with and S (at right) with such a collection of ellipses

When the ellipsoid is completely shrunken, the acceptance ratio begins to decrease. MultiNest reacts by replacing the ellipsoid with a collection of ellipsoids that contain T but overlap. T with and S with such a collection of ellipses look like those in Figure 3.2-4 (above). Subsequent samples are uniformly drawn from the union of the ellipsoids until a replacement is found. CosmoClust ([25]) introduced the idea of sampling from ellipsoids that collectively contain T to increase the acceptance ratio of the sampling process of CosmoNest. Assuming that a level set is contained by both a collection of ellipsoids and a single ellipsoid, when the (hyper-)area of the union of the ellipsoids in the collection is less than the (hyper-)area of the single ellipsoid, the collection of ellipsoids is a better proxy (than the single ellipsoid) for the level set. When the $3{,}891$st and last (for $n = 10{,}000$ samples) sample is accepted, the $9{,}997$ samples (for which the acceptance ratio is approximately $39\,\%$) and T with and S with the partially shrunken ellipses look like:

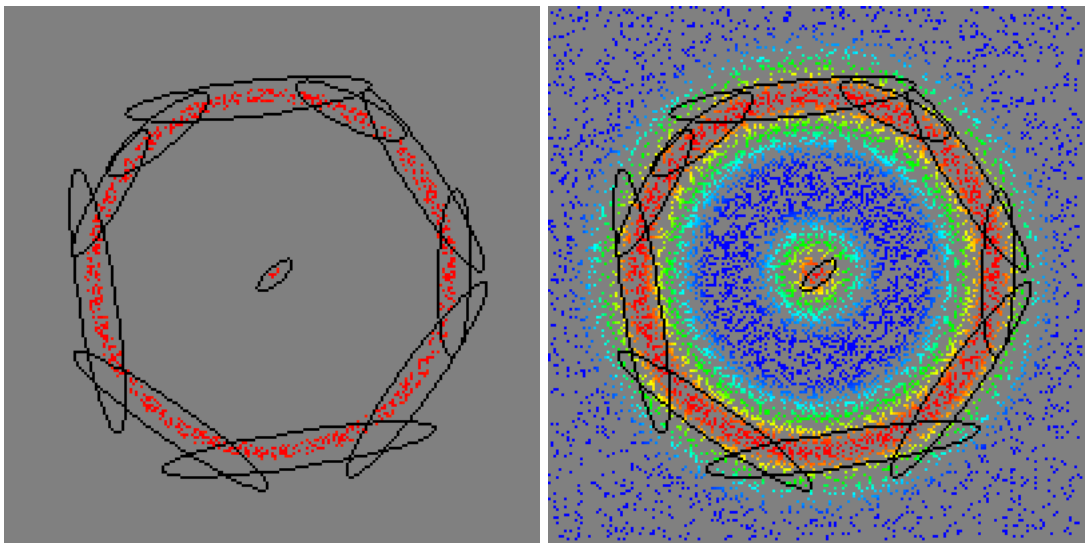


Figure 3.2-5: T (at left) with and S (at right) with the partially shrunken ellipses

The samples in T are near the top of the features. The value of the lowest-valued sample in T is approximately 0.966458, which is very high. (The value of the highest-valued sample is less than 1.0.)

The $2{,}891$ replaced samples look like:

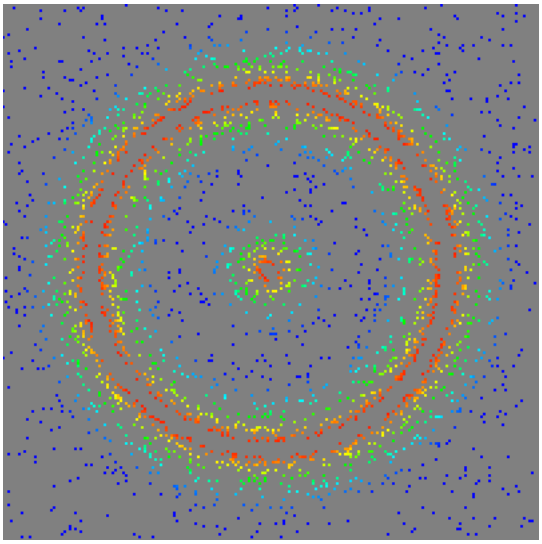


Figure 3.2-6: The $2{,}891$ replaced samples

v is computed like Nest. However, unlike Nest:

$$\tilde{E} = (A_D - W_n)v$$

That is, $\tilde{E}$ is equal to the correction to $\tilde{I}$.

MultiNest is described in more detail in [26-27]. [28] describes how MultiNest uses importance nested sampling, which is a kind of pseudo importance sampling, to compute an alternative approximate integral that uses all of the samples and supposedly is more accurate than $\tilde{I}$.

3.2.1 Demonstration

$10{,}000$ samples of the demonstration function by MultiNest look like those in Figure 3.2.1-1 (below). Note that both features are resolved but that the valley between them, which is unimportant, is sampled almost as frequently as the important regions. However, the unimportant regions near the boundaries of the domain are sampled less frequently than the important regions. If the enlargement factor were reduced, multiple ellipses would exist earlier in the sampling process and, consequently, the valley would be sampled less frequently (but still more frequently than the unimportant regions near the boundaries). However, doing so could decrease the

accuracy of the results.

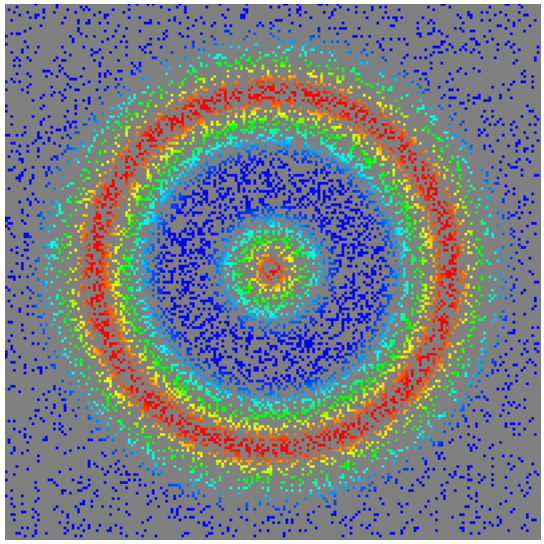


Figure 3.2.1-1: $10{,}000$ samples of the demonstration function by MultiNest

MultiNest looks promising in two dimensions but may be unable to keep its promise in higher dimensions in which very many (hyper-)ellipsoids may be necessary to cover a, say, (hyper-)spherical shell without covering much of the space that surrounds the shell.

3.2.2 Parameters

MultiNest has the following control parameters:

Name	Description	Typical Value
nest_ceff	It controls whether sampling is done with (approximately) constant efficiency by varying nest_efr.	FALSE
nest_efr	It controls the enlargement factor that corresponds to the reduction (from 1.0 to the value in (0.0, 1.0]) in sampling efficiency.	0.3
nest_IS	It controls whether pseudo importance sampling is done.	TRUE
nest_logZero	It controls which $\boldsymbol{x}$ are ignored. (They are such that $f(\boldsymbol{x})$ is less than the value.)	~-2E+308
nest_maxIter	It controls the maximum number of iterations (that is, replacements) that can be done before termination occurs. Any non-positive value is interpreted as infinity. See nest_tol.	0

Name	Description	Typical Value
nest _maxModes	It controls the maximum number of modes (that is, features) that can be integrated separately with nest_mmodal. See nest_Ztol.	10
nest_mmodal	It controls whether multimodal (that is, separate) integration is done. It is automatically set to FALSE when nest_IS is set to TRUE. See nest_Ztol and nest_maxModes.	TRUE
nest_nClsPar	It controls the number of parameters (that is, dimensions) in which to search for clusters of points (that is, modes). Where p is the value, the first p parameters are searched.	d
nest_nlive	It controls the initial value of N (that is, the number of elements in the set of the “live" samples).	1000
nest_tol	It controls the (relative) tolerance for the partial integral that remains relative to the current total integral. It is the threshold below which termination occurs. See nest_maxIter.	0.5
nest_Ztol	It controls the tolerance for a multimodal integral. It is the threshold above which a mode is integrated separately with nest_mmodal. See nest_maxModes.	-1E+90

Table 3.2.2-1: MultiNest’s control parameters

Sampling with (approximately) constant efficiency is not done in this thesis (that is, nest_ceff is set to FALSE) because doing so sacrifices accuracy for efficiency, which is unacceptable for the applications that are targeted by this thesis. Doing importance nested sampling, as in this thesis, precludes integrating modes separately (with nest_mmodal), which is unnecessary because the applications do not require multimodal integration.

3.2.3 Parallelization

During each nested sampling step, where p is the number of available processors, up to p candidate samples from the union of the ellipsoids can be drawn in parallel until at least one is acceptable. If they were, then one should be randomly accepted from among the acceptable ones (which would raise the acceptance threshold) until none are acceptable. Then, the ellipsoids are shrunk. Sampling from the ellipsoids should increase the parallel sampling efficiency as well as the regular sampling efficiency.

4 Voronoi Integration

VoroInt, which is a novel Monte Carlo method for multidimensional numerical integration, is based on Voronoi integration.

Voronoi integration is a scheme to compute $\tilde{I}$ in which w_i is the (hyper-)area of the region that is associated with $\mathbf{x}_i$ by the Voronoi decomposition of D about the $\mathbf{x}_i$, which are called *sites* in this context. Recall that the Voronoi decomposition of D about n sites is a decomposition of D into n regions such that each region contains a site and every point in D for which the distance from the point to the site is less than or equal to the distance from the point to any other site. For example, the Voronoi decomposition about $n = 10$ uniformly-distributed samples (that is, site-value pairs) of the demonstration function, where the boundaries of the regions are shown in black, looks like:

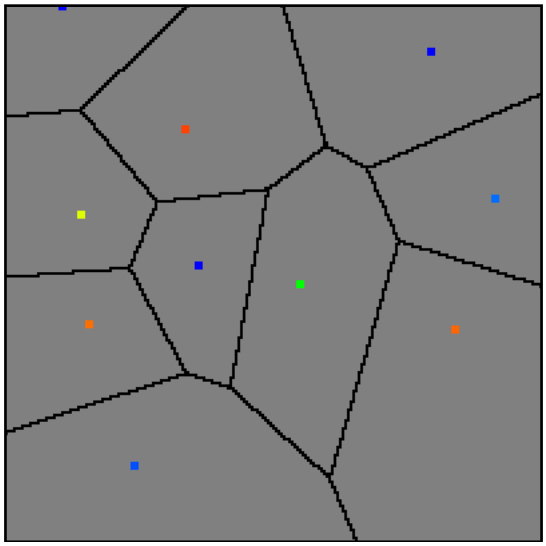


Figure 4-1: The Voronoi decomposition about $n = 10$ uniformly-distributed samples of the demonstration function

As always, the regions cover D and do not overlap, except adjacent regions, which only overlap on their common boundary. Because the boundaries are infinitesimally thin, their total (hyper-)area is equal to zero. Therefore, the sum of the (hyper-)areas of the regions is:

$$\sum_{i=1}^{n} w_i = A_D$$

Therefore, the formula for $\tilde{I}$ that the scheme uses is a Riemann sum, unlike the formulae for $\tilde{I}$ that the methods that have been described in this thesis use. PLAIN (Section 2.1) and, by extension, MISER (Section 2.3) use w_i that are independent of the samples (but total A_D). VEGAS (Section 2.2) uses w_i that approximately total A_D and Nest (Section 3.1) and MultiNest (Section 3.2) use w_i that asymptotically total A_D (but depend on the area of the region that contains $\mathbf{x}_i$). Moreover, unlike the other ways to compute $\tilde{I}$ that have been described in this thesis, the scheme is completely general. That is, the scheme can be used to compute $\tilde{I}$ using samples that are arbitrarily distributed. Therefore, the scheme can be used to compute $\tilde{I}$ using samples from any sampling process.

The scheme is precedented by, for example, VORONOI_WEIGHT ([29]), which is a method that estimates the (hyper-)areas of the regions that contain sites in the unit (hyper)cube for use (among other things) as weights in a Riemann sum. However, a definition of $\tilde{E}$ for the scheme seems unprecedented. In PLAIN/MISER/VEGAS, the approximate (absolute) error in the domain/subdomain(s)/bin(s) is the standard error about the approximate integral over the domain/subdomain(s)/bin(s). A simpler choice for the approximate error in the region that contains $\mathbf{x}_i$, $\tilde{E}_i$, is the average deviation about $w_i f(\mathbf{x}_i)$ (which is the approximate integral over the region that contains $\mathbf{x}_i$) in the neighborhood of $\mathbf{x}_i$. The “natural” choice for the neighborhood of $\mathbf{x}_i$ is the union of $\mathbf{x}_i$ and the set of the natural neighbors of $\mathbf{x}_i$, N_i, which are the sites that are contained by the regions that are adjacent to the region that contains $\mathbf{x}_i$. Thus:

$$\tilde{E}_i = w_i \frac{\sum_{\mathbf{x}_j \in N_i} f(\mathbf{x}_j) - f(\mathbf{x}_i)}{1 + |N_i|}$$

$\mathbf{x}_i$ is omitted from the sum because the summand for which $\mathbf{x}_j = \mathbf{x}_i$ always equals zero. In MISER/VEGAS, the total approximate (absolute) error in the subdomain(s)/bin(s) is the square root of the sum of the squares of the approximate (absolute) error(s) in the subdomain(s)/bin(s). A simpler choice for $\tilde{E}$ is:

$$\tilde{E} = \sum_{i=1}^{n} \tilde{E}_i$$

Described in the rest or this chapter is VoroInt (in Section 4.1), which is a novel method based on the scheme that uses the $\tilde{E}_i$ to efficiently reduce $\tilde{E}$.

4.1 VoroInt

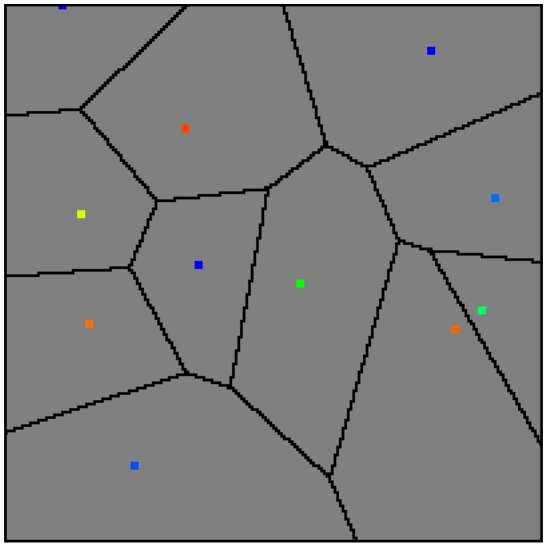


Figure 4.1-1: The example Voronoi decomposition, after the region for which $\tilde{E}_i$ is maximum is sampled

VoroInt is a novel method based on Voronoi Integration that uses the $\tilde{E}_i$ to direct a sampling process that efficiently reduces $\tilde{E}$. A sensible strategy to efficiently reduce $\tilde{E}$ is to reduce a maximal $\tilde{E}_i$ by sampling the associated

region(s). Therefore, a region for which $\tilde{E}_i$ is maximal should be sampled next. The example Voronoi decomposition, after the (bottom-right) region for which $\tilde{E}_i$ is maximum is sampled, looks like that in Figure 4.1-1 (above). The new sample site is contained by the new (triangular) region. A new region steals (hyper-)area from the regions adjacent to it and may separate former adjacent regions. Therefore, where $\mathbf{x}_n$ is the new site, when $\mathbf{x}_n$ is inserted into the Voronoi decomposition, N_n, w_n, and $\tilde{E}_n$ must be set and, for $\mathbf{x}_i \in N_n$, N_i, w_i, and $\tilde{E}_i$ must be reset so that the sampling process can continue. Furthermore, $\tilde{I}$ and $\tilde{E}$ must be updated so that the method can continue.

Initially, $(\mathbf{x}_n, f(\mathbf{x}_n))$ is chosen from a set of pre-existing samples called *hints*. The hints are chosen from the set without replacement until the set is empty. They can be used to uniformly search D (like the 10 initial samples in the example), guide the sampling process to important features that are known before the current run, or continue a previous run. If the method were (forced) to terminate immediately after the hints were processed, then it could be used to compute an alternate $\tilde{I}$ and $\tilde{E}$ for samples from any other method. When no hints are given, $\mathbf{x}_1$ is drawn from D.

The algorithm for the method, where D, f, H, which is the set of hints, and one or more of $\mathrm{tol}_{\mathrm{abs}}$, which is the absolute error tolerance, $\mathrm{tol}_{\mathrm{rel}}$, which is the relative error tolerance, $n_{\min}$, which is the minimum number of samples, and $n_{\max}$, which is the maximum number of samples, are given, is:

$\tilde{I} \leftarrow 0.0$

$\tilde{E} \leftarrow \infty$

$n \leftarrow 0$

While $\tilde{E} \geq \mathrm{tol}_{\mathrm{abs}}$ and $\tilde{E} \geq \mathrm{tol}_{\mathrm{rel}}\tilde{I}$ and ($n < n_{\min}$ or $n_{\max} \leq 0$ or $n < n_{\max}$):

$n \leftarrow n + 1$

If $H = \emptyset$:

If $n > 1$:

Draw $\mathbf{x}_n$ from a region associated with $\max_{1 \leq i < n} \tilde{E}_i$.

Else:

Draw $\mathbf{x}_1$ from D.

Evaluate f at $\mathbf{x}_n$.

Else:

Choose $(\mathbf{x}_n, f(\mathbf{x}_n))$ from H without replacement.

Insert $\mathbf{x}_n$ into the Voronoi decomposition.

Set $\tilde{E}_n$.

For $\mathbf{x}_i \in N_n$:

Reset $\tilde{E}_i$.

Reset $\tilde{I}$, $\tilde{E}$.

The details of drawing $\mathbf{x}_n$ from a region and inserting $\mathbf{x}_n$ into the Voronoi decomposition follow.

Drawing $\mathbf{x}_n$ from a Region

Each region has a bounding box that is set or reset when the region is created or modified. To draw $\mathbf{x}_n$ from the region that contains $\mathbf{x}_i$, candidates for $\mathbf{x}_n$ are drawn from the bounding box of the region until a candidate is drawn from the region (that is, no farther from $\mathbf{x}_i$ than from any other site). To decide whether to accept or reject a candidate, distances only need to be computed between the candidate and the sites in $\{x_i\} \cup N_i$, which define a local Voronoi decomposition of D about $\mathbf{x}_i$. The local Voronoi decomposition that is used while drawing $\mathbf{x}_n$ from the region of the example for which $\tilde{E}_i$ is maximum, where the candidates are shown in white and the bounding box is shown in light gray, looks like:

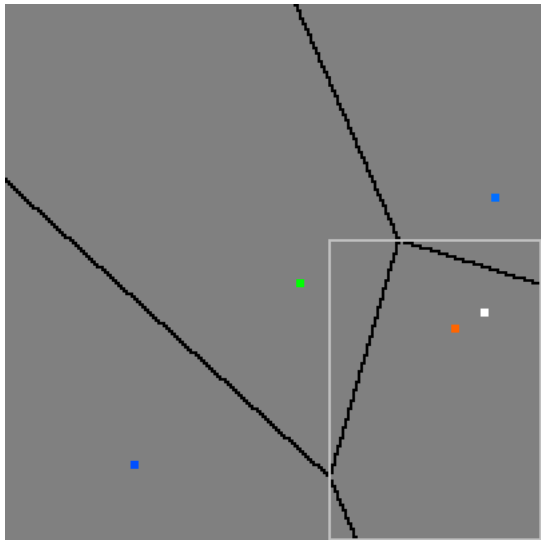


Figure 4.1-2: The local Voronoi decomposition that is used while drawing $\mathbf{x}_n$ from the region of the example for which $\tilde{E}_i$ is maximum

The first (and only) candidate is in the region. The local Voronoi decomposition is used because $1 + |N_i|$ is usually much less than n and the computational complexity of nearest-neighbor algorithms increases with the number of sites that are input. Furthermore, the bounding box is used because the area of the bounding box is usually much less than A_D and the number of candidates is expected to be directly proportional to the ratio of the area from which they are drawn to the area of the region.

The algorithm for drawing $\mathbf{x}_n$ from a region, where $\mathbf{x}_s$ is the site contained by the region, is:

Loop:

Draw $\mathbf{x}_n$ from the bounding box of the region

If $\mathbf{x}_n$ is no farther from $\mathbf{x}_s$ than $\mathbf{x}_n$ is from $\mathbf{x}_i \in N_n$:

Break

Inserting $\mathbf{x}_n$ into the Voronoi Decomposition

After $\mathbf{x}_n$ is drawn, N_n must be set before w_n and, subsequently, $\tilde{E}_n$ can be set. Furthermore, for $\mathbf{x}_i \in N_n$, N_i must be reset before w_i and, subsequently,

$\tilde{E}_i$ can be reset. To set N_n and reset, for $\mathbf{x}_i \in N_n$, N_i, an algorithm such as Quickhull must be used. Qhull is used to determine whether a pair of m sites are neighbors. Because the space and time complexity of Quickhull is $O(m^{\lceil d/2 \rceil})$, m should be as small as possible. Therefore, the global Voronoi decomposition about n sites must be avoided. Crucially, N_n can be set and, for $\mathbf{x}_i \in N_n$, N_i can be reset by way of local Voronoi decompositions about $m \ll n$ (when n is large) sites. The (final) local Voronoi decomposition that is used while inserting $\mathbf{x}_n$ into the example Voronoi decomposition, looks like:

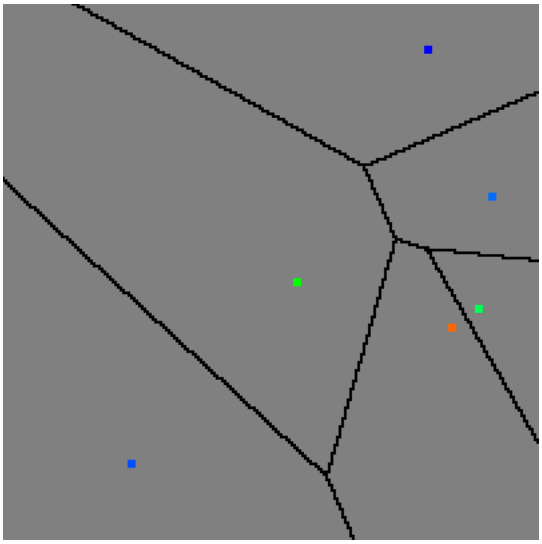


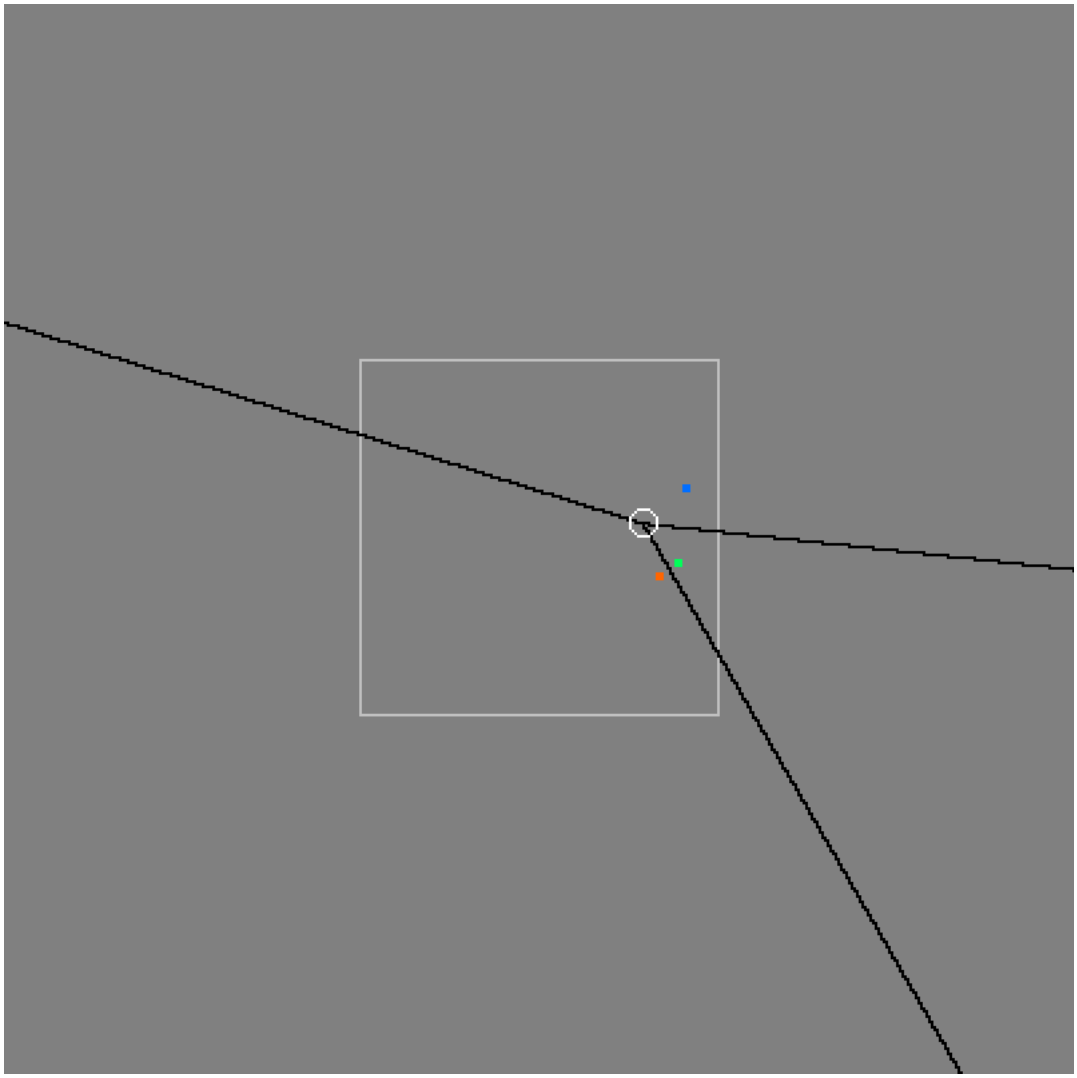
Figure 4.1-3: The local Voronoi decomposition that is used while inserting $\mathbf{x}_n$ into the example Voronoi decomposition

The $m = 6$ sites that are necessary to reset, for $\mathbf{x}_i \in N_n$, N_i are selected by iteratively adding $\mathbf{x}_i \in N_n$ to the input of Qhull, which starts with $\{\mathbf{x}_n\}$, and updating N_n from the output of Qhull for $\mathbf{x}_n$ until N_n stops changing and is, therefore, correct. Then, for $\mathbf{x}_i \in N_n$, N_i is updated from the output of Qhull for $\mathbf{x}_i$. The aforementioned output of Qhull contains the adjacency relationships between the sites that are input, which, where S is a set of sites, is written as $\mathrm{Adjacency}(S)$ hereafter. Qhull also outputs the extreme vertices of the regions that are defined by the sites that are input, which, where S is a set of sites and $s \in S$ is contained by the region of interest, is

written as $\mathrm{Vertices}(S, s)$ hereafter. The extreme vertices are used to set w_n and, for $\mathbf{x}_i \in N_n$, reset w_i and to create and modify bounding boxes. A weight is the volume that is output by Qhull for the region that is bounded by the extreme vertices that are input, which, where V is a set of vertices, is written as $\mathrm{Volume}(V)$ hereafter.

To ensure that the extreme vertices that are output by $\mathrm{Vertices}(S, s)$ bound the region of interest (that contains s) with respect to D, the reflection of s through each boundary of D is added to S, as is done in [30] and shown below.

Before



After

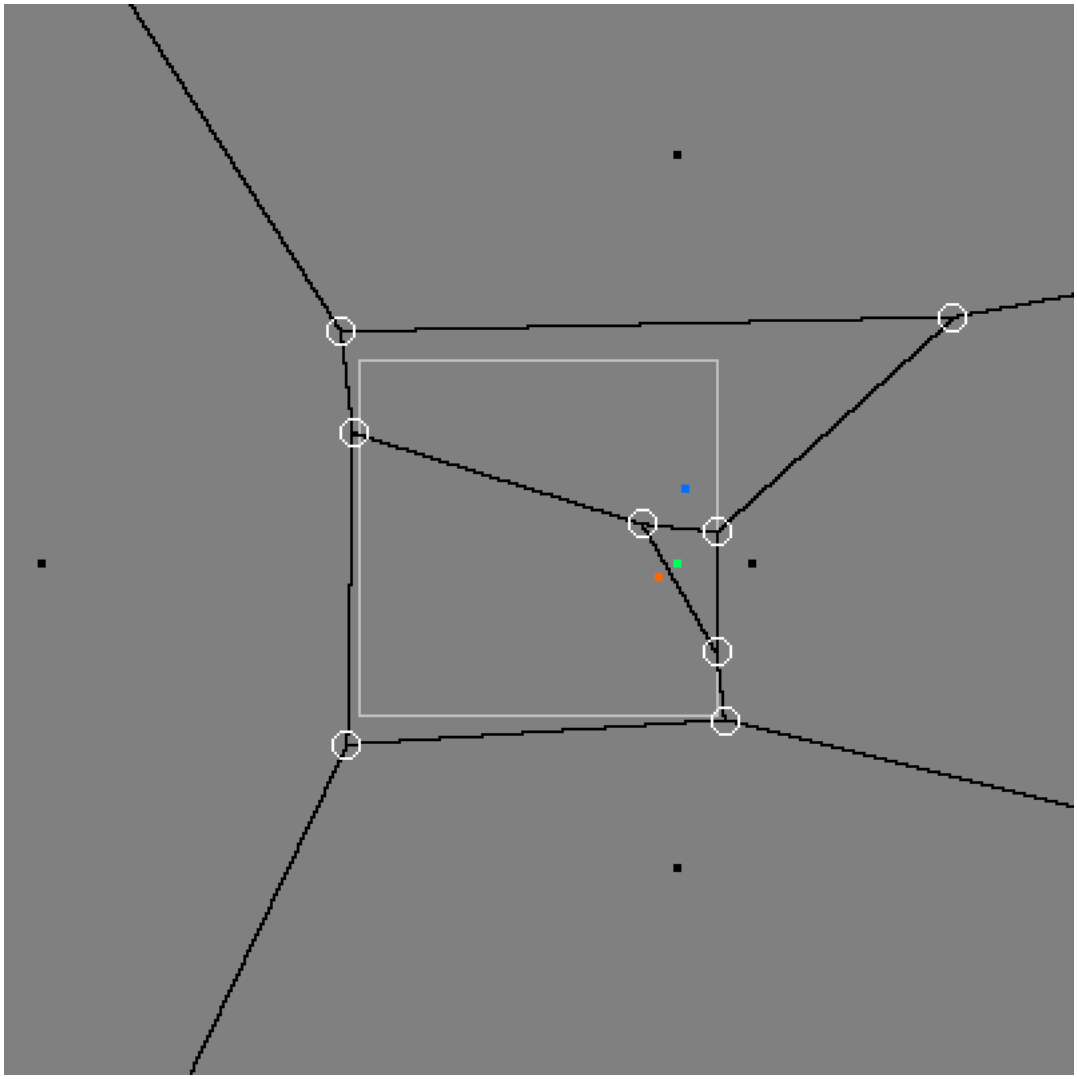


Figure 4.1-4: The extreme vertices and the boundaries of the new region look before and after the reflections of $\mathbf{x}_n$ through the boundaries of $[-0.5,0.5]^2$ are added to the local Voronoi decomposition about $\mathbf{x}_n$

Consider how the extreme vertices and the boundaries of the new region (that contains $\mathbf{x}_n$) look before and after the reflections of $\mathbf{x}_n$ through the boundaries of $[-0.5,0.5]^2$ are added to the local Voronoi decomposition about $\mathbf{x}_n$, where the boundaries of $[-0.5,0.5]^2$ are underlaid in light gray, the extreme vertices are circled in white, and the reflections are shown in black as in Figure 4.1-4 (above). The new region becomes bound courtesy of the reflection of $\mathbf{x}_n$ through the right boundary of $[-0.5,0.5]^2$. Some of the reflections may be unnecessary to bound the region of interest with

respect to D. However, because the identification of only the necessary reflections is onerous, all of the reflections are added. After a region is bound, its volume and bounding box can be computed.

The algorithm for inserting $\mathbf{x}_n$ into the Voronoi decomposition of D about $\{\mathbf{x}_i\}_{1 \le i < n}$, where $\mathbf{x}_s$ is the site contained by the region whence $\mathbf{x}_n$ was drawn or placed (if $\mathbf{x}_n$ was drawn from H) or null if $\mathbf{x}_n$ was drawn from D, is:

$c \leftarrow 0$

$N_n \leftarrow \{\mathbf{x}_s\}$

While $c \neq |N_n|$:

$S \leftarrow \{\mathbf{x}_n\} \cup N_n \cup (\cup_{\mathbf{x}_i \in N_n} N_i)$

$\{N_i'\}_{x_i \in S} \leftarrow \text{Adjacency}(S)$

$c \leftarrow |N_n|$

$N_n \leftarrow N_n'$

Let R be the set of reflections of $\mathbf{x}_n$ through each boundary of D.

$V \leftarrow \text{Vertices}(\mathbf{x}_n \cup N_n \cup R, \mathbf{x}_n)$

$w_n \leftarrow \text{Volume}(V)$

Create the bounding box for the region that contains $\mathbf{x}_n$ using V.

For $\mathbf{x}_i \in N_n$:

$N_i \leftarrow N_i'$

Let R be the set of reflections of $\mathbf{x}_i$ through each boundary of D.

$V \leftarrow \text{Vertices}(\mathbf{x}_i \cup N_i \cup R, \mathbf{x}_i)$

$w_i \leftarrow \text{Volume}(V)$

Modify the bounding box for the region that contains $\mathbf{x}_i$ using V.

4.1.1 Demonstration

$10{,}000$ samples of the demonstration function by VoroInt look like:

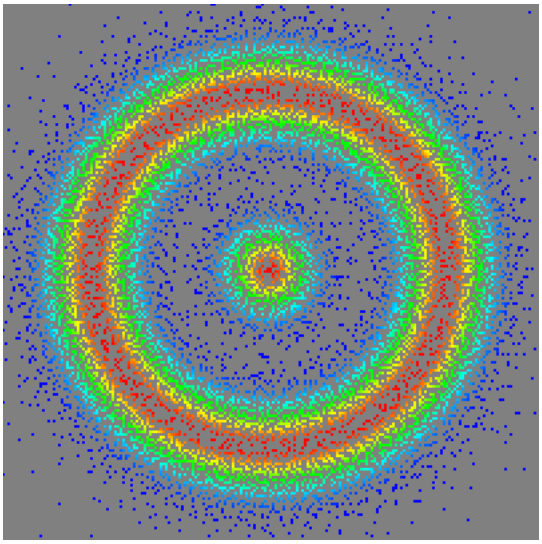


Figure 4.1.1-1: $10{,}000$ samples of the demonstration function by VoroInt

Note both features are resolved but the unimportant (that is, low-valued or blue) regions, including the region between the features, and the important (that is, high-valued or red) regions are sampled infrequently because VoroInt samples by gradient rather than by importance. Compare the samples with the following graph of the magnitude of the gradient of the demonstration function in which the magnitudes have been normalized by the maximum magnitude:

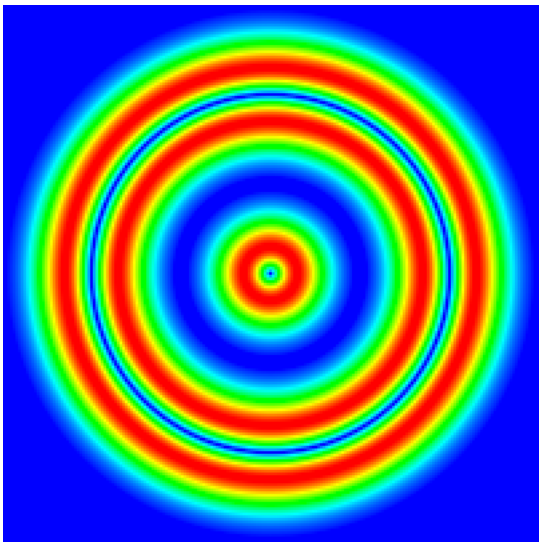


Figure 4.1.1-2: The magnitude of the gradient of the demonstration function

VoroInt should and does avoid sampling flat (that is, low-gradient) regions because they cannot be further resolved. VoroInt samples sloped (that is, high-gradient) regions because they can be further resolved and may lead to undiscovered features.

4.1.2 Parameters

VoroInt has the following control parameters:

Name	Description	Default Value
$\mathrm{tol}_{\mathrm{abs}}$	It controls the absolute tolerance for the error. It is the threshold below which termination occurs. See $\mathrm{tol}_{\mathrm{rel}}$.	0.0
$\mathrm{tol}_{\mathrm{rel}}$	It controls the relative tolerance for the error. It is the threshold below which termination occurs. See $\mathrm{tol}_{\mathrm{abs}}$.	0.0
$\mathrm{n}_{\mathrm{min}}$	It controls the minimum number of samples. It is the threshold below which termination cannot occur and used to avoid premature termination. See $\mathrm{n}_{\mathrm{max}}$.	0
$\mathrm{n}_{\mathrm{max}}$	It controls the maximum number of samples. It is the threshold at which termination occurs and used with $\mathrm{tol}_{\mathrm{abs}}$ or $\mathrm{tol}_{\mathrm{rel}}$ to avoid an unreasonably long run or without them to obtain exactly $\mathrm{n}_{\mathrm{max}}$ samples. See $\mathrm{n}_{\mathrm{min}}$.	0

Table 4.1.2-1: VoroInt's control parameters

Note the parameters only control termination.

4.1.3 Parallelization

During the insertion of $\mathbf{x}_n$, the c neighbors of $\mathbf{x}_n$ can be updated in parallel by way of straightforward parallelization of the inner for loop. If they were, where p is the number of available processors, then the cost of updating the neighbors would be reduced by a factor of $1/\min(p, c)$, which is limited by c (which is the number of iterations of the loop). The parallelism increases as d increases because, as is shown in Section 5.4.2, c increases with d.

Alternatively, the algorithm can be (additionally) parallelized in a potentially more effective way. Because the insertion of a site into the Voronoi decomposition does not require the value of f at the site to be known until the

approximations of the integral and error are updated, the evaluation of the site could be done in the background of its insertion. Moreover, t sites could be drawn from the top t highest-priority regions and evaluated in the background of their collective insertions by way of replacement of the body of the outer while loop with the following:

For j from 1 to t:
    If $H = \varnothing$:
        If $n > 0$:
            Draw $\mathbf{x}_{n+j}$ from the region with priority #j.
        Else:
            Draw $\mathbf{x}_j$ from D.
        Evaluate f at $\mathbf{x}_{n+j}$ **in the background**.
    Else:
        Choose $(\mathbf{x}_{n+j}, f(\mathbf{x}_{n+j}))$ from H without replacement.
For j from 1 to t:
    Insert $\mathbf{x}_{n+j}$ into the Voronoi decomposition.
For j from 1 to t:
    Set $\tilde{E}_{n+j}$.
    For $\mathbf{x}_i \in N_{n+j}$:
        Reset $\tilde{E}_i$.
Reset $\tilde{I}$, $\tilde{E}$.
$n \leftarrow n + t$

Assuming all of the features of f would be discovered by VoroInt regardless of t, VoroInt would converge to the same value regardless. However, if a sample is drawn from a region that is not the top priority but eventually would have been so, then the rate of convergence of VoroInt would be greater than

it would have been otherwise. The rate of convergence would be least when only the top-priority region must be sampled (for example, when the only unresolved feature is a spike) and greatest when the top t highest-priority regions must be sampled (for example, when there are at least t unresolved spikes).

4.1.4 Computational Complexity

The computational complexity of VoroInt is dominated by the complexity of the insertion of $\mathbf{x}_n$ into the Voronoi decomposition, in which the significant subcomputations are the while loop and the for loop. Let l be the number of iterations of the while loop, which is one greater than the greatest degree of separation between $\mathbf{x}_s$ and any element of N_n. The number of iterations of the for loop is $c = |N_n|$. The typical values of l and c are empirically determined in Section 5.4.2.

In the while loop, the call to $\mathrm{Adjacency}$ is the dominant operation. The most significant call of $\mathrm{Adjacency(S)}$ is the last call because $|S|$ increases. Therefore, the cost of the while loop is at most l times the cost of the last call to $\mathrm{Adjacency}$. Assuming every site has c neighbors (which is not necessarily true), for the last call, $|S| \leq c^2$ because, in the case of equality, S contains all of the c neighbors of all of the c sites in N_n and none of the sites have a mutual neighbor. The case of equality is rare because some of the sites usually have mutual neighbors. The typical value of $|S|$ may be closer to c than to c^2. Therefore, the complexity of the while loops is less than or equal to l times the complexity of Qhull with c^2 sites, which is $O(lc^{2\lceil d/2 \rceil})$.

In the for loop, the call to $\mathrm{Vertices}$ is the dominant operation (because it is performed in a space with greater dimensionality than that in which Volume

is performed). The call before the loop can be ignored because $O(1+c) = O(c)$. Again assuming every site has c neighbors, Vertices operates on $1+c+2d$ sites, which asymptotically is $O(1+c+2d) = O(c+2d)$ sites. Therefore, the complexity of the for loops is equal to c times the complexity of Qhull with $O(c+2d)$ sites, which is $O(c(c+2d)^{\lceil d/2 \rceil})$.

Therefore, the complexity of the insertion of $\mathbf{x}_n$ is $O(lc^{2\lceil d/2 \rceil} + c(c+2d)^{\lceil d/2 \rceil})$. So, the complexity of VoroInt, which is that of n insertions, is $O(n(lc^{2\lceil d/2 \rceil} + c(c+2d)^{\lceil d/2 \rceil}))$.

5 Results

In this chapter, where n is the number of evaluations of the integrand, the accuracy versus n of the Monte Carlo integration methods PLAIN, VEGAS, and MISER (which are described in Chapter 2) and the nested sampling methods Nest and MultiNest (which are described in Chapter 3) are compared to that of the novel Voronoi integration method VoroInt (which is described in Chapter 4) for various integrands. Also, for each integrand, the average and maximum counts of the significant subcomputations of VoroInt, from which the computational complexity of VoroInt is estimated, are reported.

The integrands are the demonstration function and its inner and outer features on $D = [-0.5,0.5]^d$. The inner feature is the following Gaussian function:

$$G(\mathbf{x}) = e^{-0.5\left(\frac{\sqrt{x_1^2 + x_2^2 + \cdots + x_d^2}}{0.05}\right)^2}$$

Its integral is:

$$I_G(d) = \left(0.05\sqrt{2\pi}\right)^d$$

The outer feature is the following Gaussian circle function:

$$C(\mathbf{x}) = e^{-0.5\left(\frac{\sqrt{x_1^2 + x_2^2} - 0.\overline{3}}{0.05}\right)^2}$$

Its integral ([31]) is:

$$I_C = 0.262449$$

Recall (from Section 1.2) the features are relevant because the former is a spike and the latter is a degeneracy the likes of which are common in cosmology.

The significant subcomputations of VoroInt are the iterations of the while and

for loops that are executed when it inserts a site into a Voronoi decomposition. Recall (from Section 4.1) each iteration of the while loop involves a call to Qhull for adjacency information and each iteration of the for loop involves a call to Qhull for the vertices of a region and a subsequent call to Qhull for the volume of the region.

The actual accuracy of a method is measured by E/I (that is, E relative to I). The estimated accuracy of a method is measured by $\tilde{E}/\tilde{I}$ (that is, $\tilde{E}$ relative to $\tilde{I}$), which is usually used as the termination criterion for the method. If $\tilde{E}/\tilde{I}$ for a method were different from E/I for the method, then the method could terminate early or late and, consequently, output inaccurate results or waste evaluations. For VoroInt, $\tilde{E}/\tilde{I}$ can have the opposite sign of $\tilde{I}$ because $\tilde{E}$ is signed. Whereas, for the other methods that are considered herein, $\tilde{E}/\tilde{I}$ has the same sign as $\tilde{I}$ because $\tilde{E}$ is unsigned. For MultiNest without importance sampling, $\tilde{E}/\tilde{I}$ is unavailable until termination because $\tilde{E}$ is unavailable until then.

PLAIN outputs inf(inity) for $\tilde{E}$ for $n = 1$. Rather than output inf, MISER requires $n \geq 2$ and, therefore, produces no output for $n < 2$. Similarly, VEGAS requires, where k is the number of iterations, $n \geq 2k$ and, therefore, produces no output for $n < 10$. MultiNest produces no output for, where N is the number of samples in its working set, $n < N$ or for the samples that are unaccepted into its working set, despite their effect on the results of importance nested sampling. For each group of iterations of VEGAS (that is, $n = 1..5, n = 6..10, \cdots$), the weighted averages of the outputs of the iterations are used for each iteration in the group. For each sample that is not accepted into the working set of MultiNest, the output for the last sample that was accepted into the working set (which may not have been output had a hard-coded value not been modified) is used. For MISER, the output varies much between successive values of n after the first stratification

because the separate runs that correspond to the values become uncorrelated then. Before then, the output is identical to that of PLAIN. Nest produces output for all of its samples, but effectively outputs the same value for $\tilde{E}$ for the last N samples.

In the plots of accuracy versus n, E/I for a method is shown in red, green, blue, or dark gray and $\tilde{E}/\tilde{I}$ for the method is shown in the complementary color (that is, cyan for red, magenta for green, yellow for blue, and light gray for dark gray). Points that represent $\tilde{E}/\tilde{I}$ for VEGAS, which are shown in yellow, are outlined in black only if the results of the group of iterations that correspond to the points were consistent (that is, for the results, $|\chi^2/5 - 1| < 0.5$). The plots are in pairs of a plot that shows VoroInt and the Monte Carlo integration methods and a plot that shows VoroInt and the nested sampling methods. In the plots that show the nested sampling methods, MultiNest with importance sampling is called MultiNest_IS.

5.1 Demonstration Function

The demonstration function is $G(\mathbf{x}; 2) + C(\mathbf{x})$. Therefore, in this section:

$$I = I_G(2) + I_C = 0.278157$$

The function is integrated by the methods using $n = 10{,}000$ samples. VoroInt was forced to sample the integrand exactly $10{,}000$ times, using $n_{\min} = n_{\max} = 10{,}000$. For MultiNest, the number of samples cannot be precisely chosen. Instead, the number of replacements, which is greater than or equal to the number of samples, is set to n.

5.1.1 Accuracy

The following plots of accuracy versus n show the points that correspond to the samples in the demonstrations of the methods (in Sections 2.1.1, 2.2.1, 2.3.1, 3.1.1, 3.2.1, and 4.1.1). Because some of the methods converge in $n \ll 10{,}000$ samples, the points are shown for $n = 1..100$, $n = 1..1{,}000$,

and $n = 1..10{,}000$.

The first 100 points for VoroInt and other Monte Carlo methods are shown in Figure 5.1.1-1a (below). The first 100 points for VoroInt (again) and nested sampling methods are shown in Figure 5.1.1-1b (below). Recall (from Chapter 3) $N = 10$ was used with Nest and $N = 1000$ was used with MultiNest. VEGAS' E/I initially is -1, which indicates its $\tilde{I}$ initially is 0 and, thereby, why its $\tilde{E}/\tilde{I}$ initially is not shown. VEGAS takes many samples to start because each of its independent iterates is allocated only 1 out of every 5 samples. Note the consistency of VEGAS' intermediate results does not guarantee the accuracy of its results. For example, at $n = 100$, VEGAS' $\tilde{E}/\tilde{I}$ is approximately 0.3 but the magnitude of its E/I is approximately 0.7. MISER's (or PLAIN's) $\tilde{E}/\tilde{I}$ well approximates its E/I, which is fairly accurate after approximately $n = 50$. VoroInt's $\tilde{E}/\tilde{I}$ is about 0 when its E/I is not (for example, at $n = 0$). Had $n_{\min}$ been unused and $\mathrm{tol}_{\mathrm{abs}}$ or $\mathrm{tol}_{\mathrm{rel}}$ been used, VoroInt would have terminated prematurely. Therefore, $n_{\min}$ should be used unless sufficient hints are provided to prevent premature termination of VoroInt. Nest rapidly converges to a solution that is almost 0.2 too large. Had $N > 10$ been used with Nest, it probably would have converged less rapidly to a better solution. If MultiNest were to produce output for $n < N$, the output would be similar to that of PLAIN because MultiNest initially samples in the same way as PLAIN.

The first $1{,}000$ points for VoroInt and other Monte Carlo methods are shown in Figure 5.1.1-2a (below). The first $1{,}000$ points for VoroInt and nested sampling methods are shown in Figure 5.1.1-2b (below). For $n > 250$, VoroInt's and the Monte Carlo integration methods' E/I and $\tilde{E}/\tilde{I}$ are within 0.1 of 0. However, VEGAS' intermediate results are inconsistent for $n = 361..640$.

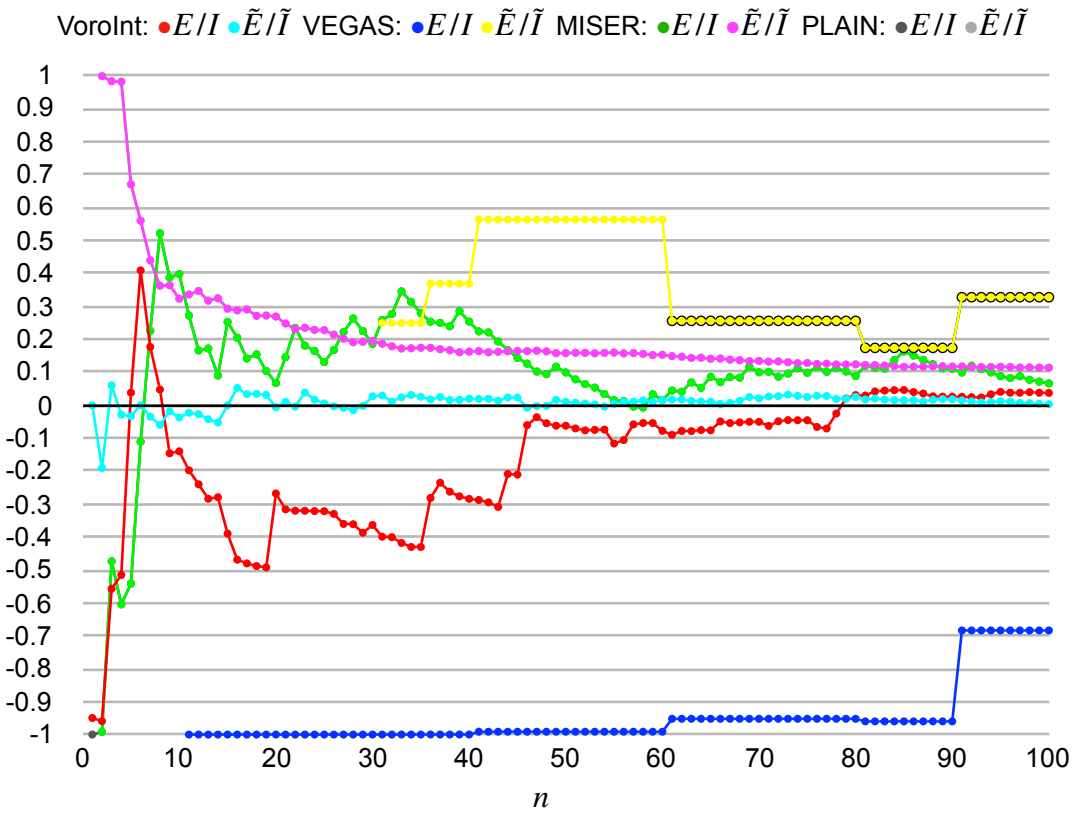


Figure 5.1.1-1a: The first 100 points for VoroInt and other Monte Carlo methods

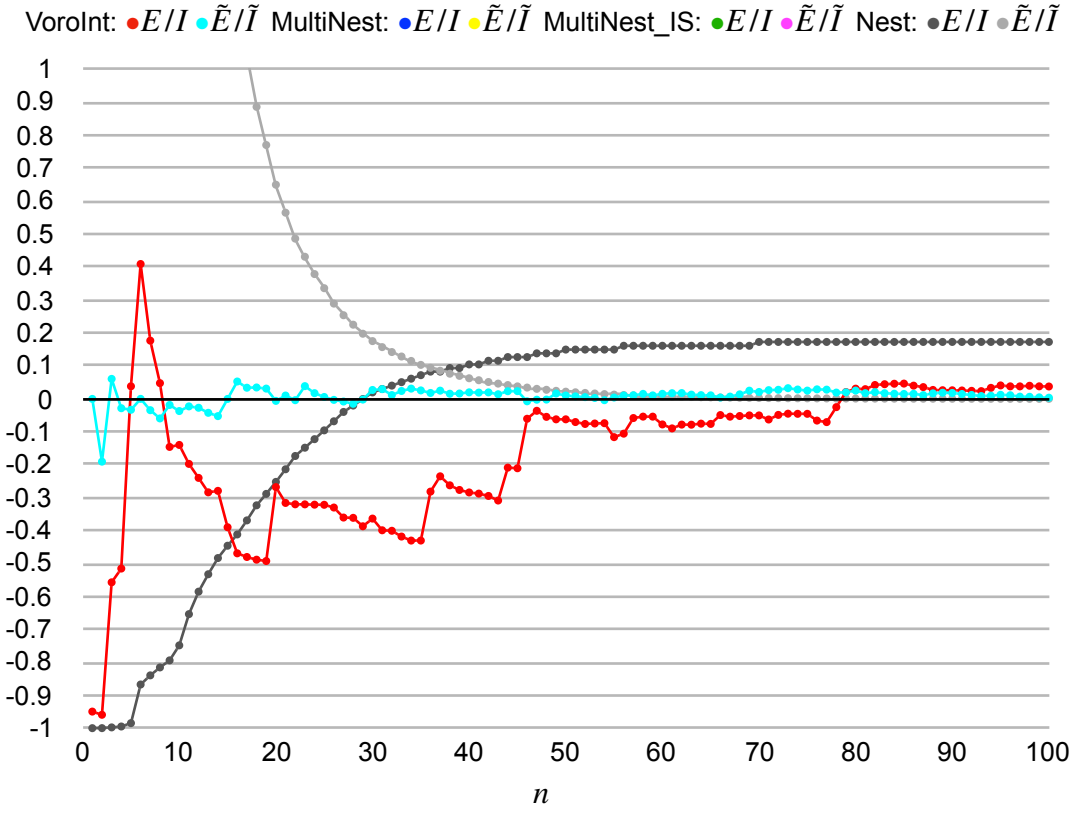


Figure 5.1.1-1b: The first 100 points for VoroInt and nested sampling methods

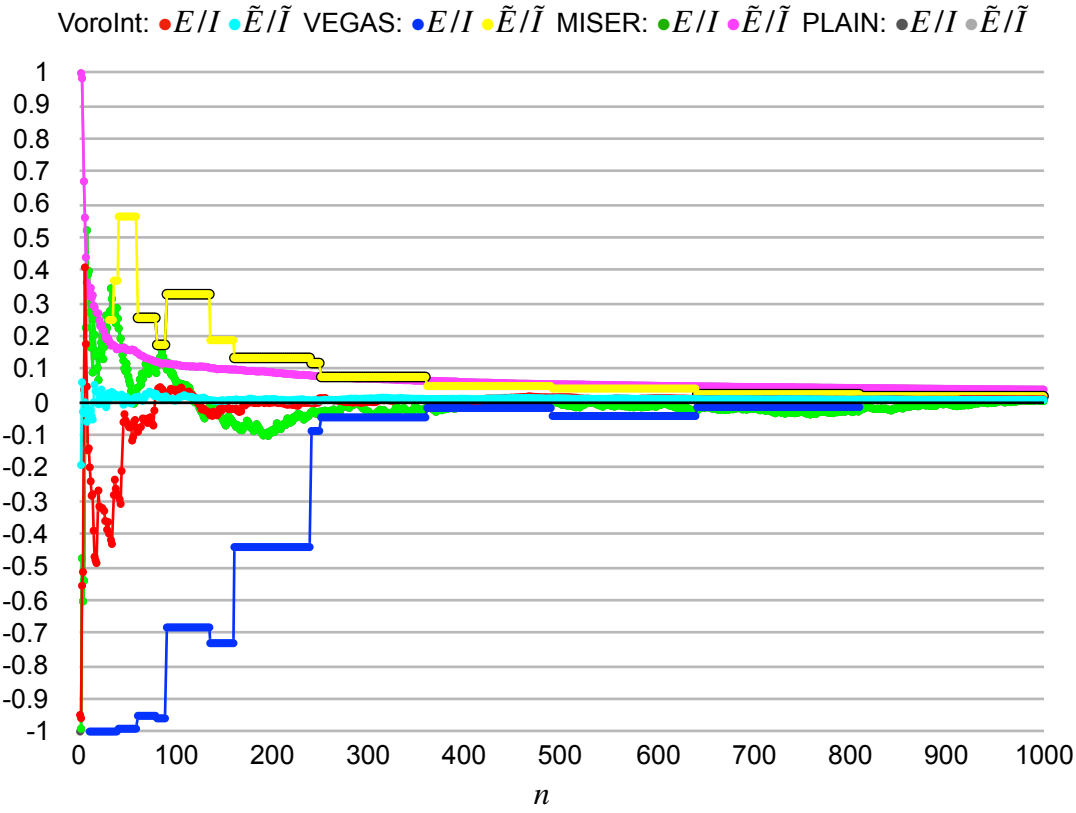


Figure 5.1.1-2a: The first $1{,}000$ points for VoroInt and other Monte Carlo methods

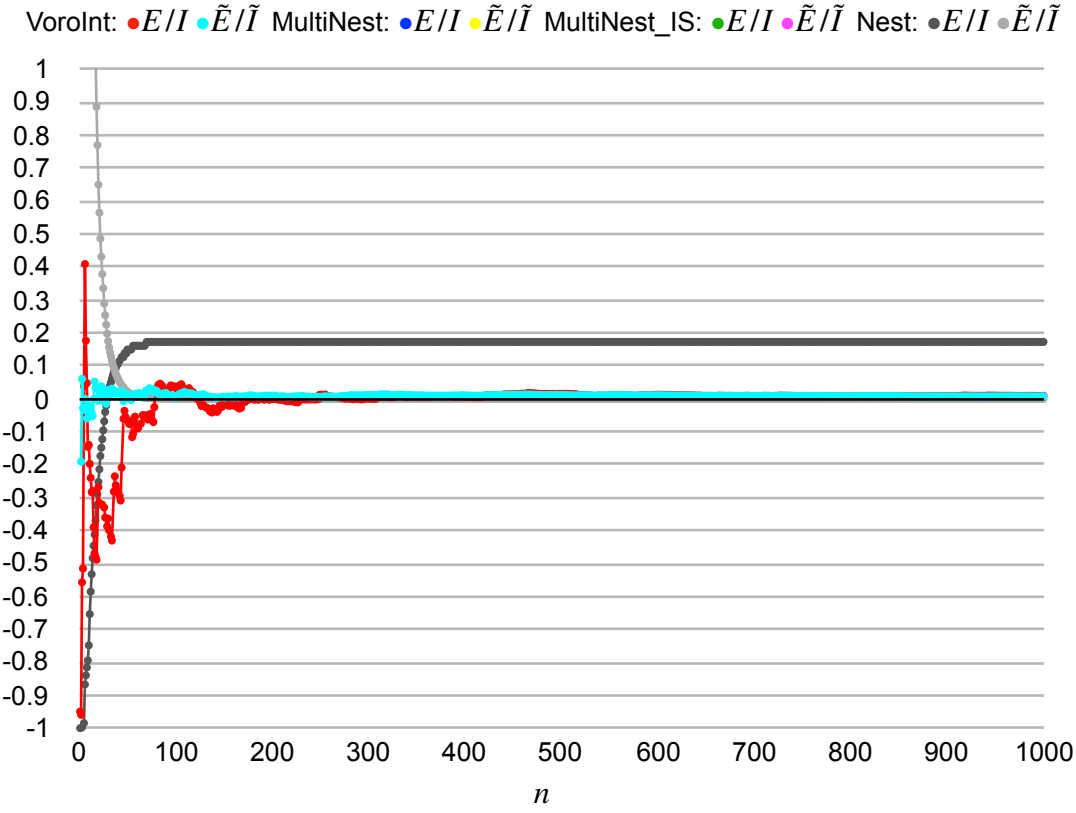


Figure 5.1.1-2b: The first $1{,}000$ points for VoroInt and nested sampling methods

Finally, all $10{,}000$ points for VoroInt and other Monte Carlo methods on a smaller scale are:

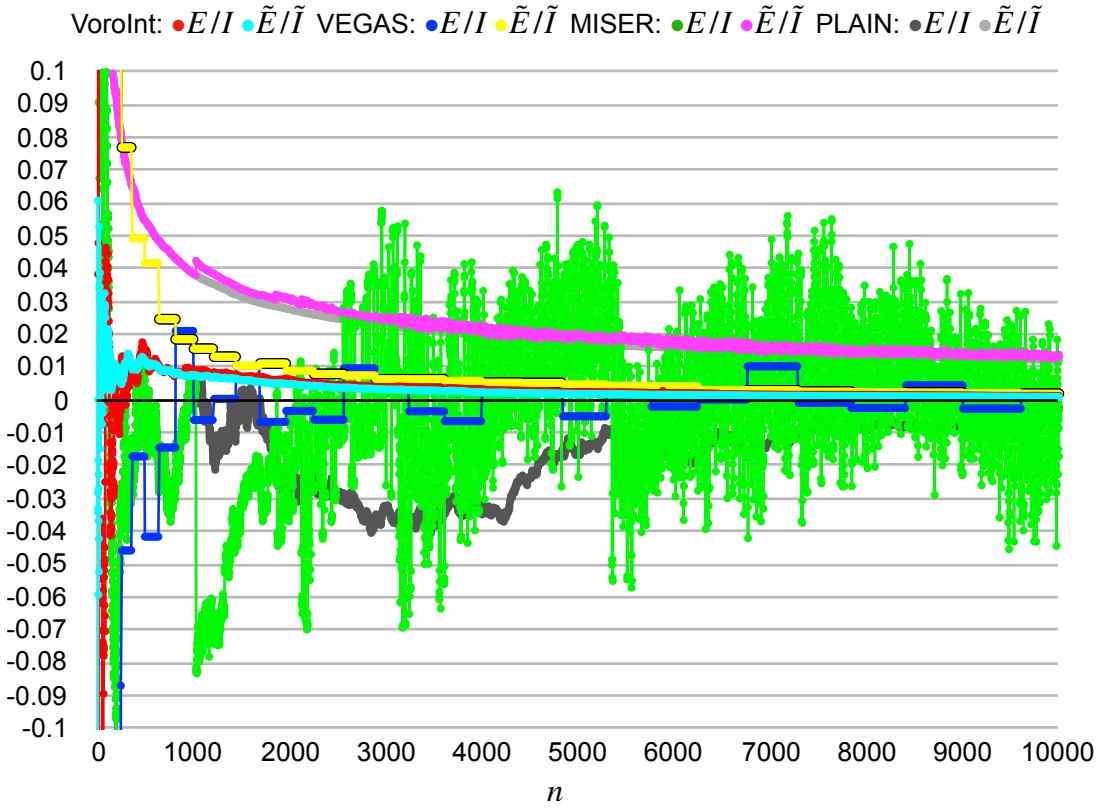


Figure 5.1.1-3a: All $10{,}000$ points for VoroInt and other Monte Carlo methods

MISER first stratifies at $n = 1{,}000$ and, consequently, separates from PLAIN. After the first stratification, MISER's $\tilde{E}/\tilde{I}$ becomes erratic but remains tightly bound, whereas its E/I becomes erratic and loosely bound, albeit about 0. PLAIN's $\tilde{E}/\tilde{I}$ exhibits the classic Monte Carlo $1/\sqrt{n}$ behavior that is characterized by a long tail. VEGAS' E/I intermittently moves away from 0, sometimes with consistent intermediate results (for example, about $n = 7{,}000$), but, nonetheless, is tightly bound about 0 by $n = 10{,}000$, as is its $\tilde{E}/\tilde{I}$. After the large rise of VoroInt's E/I at approximately $n = 500$, which may correspond to the detection of $G(\mathbf{x})$, VoroInt's E/I and $\tilde{E}/\tilde{I}$ align and fall together. Moreover, the magnitude of VoroInt's E/I is usually smaller than those of the other methods. If not for the intermittent moves, the magnitude of VEGAS' E/I would be similar to that of VoroInt.

Finally, all $10{,}000$ points for VoroInt and nested sampling methods on the original scale are:

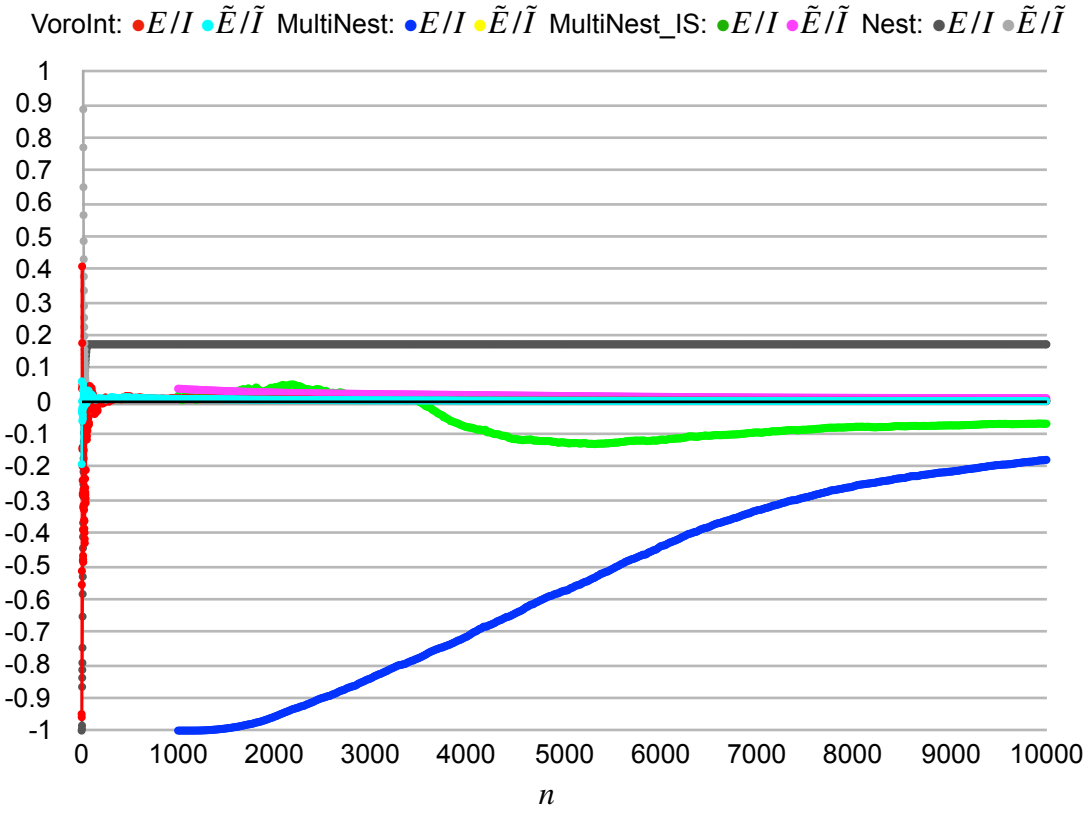


Figure 5.1.1-3b: All $10{,}000$ points for VoroInt and nested sampling methods

MultiNest(_IS) first produces output at $n = 1{,}000$. MultiNest's E/I monotonically increases from -1 (that is, nothing yet detected) to approximately -0.2 at $n = 10{,}000$ and would continue to increase beyond $n = 10{,}000$ because it had not yet converged. It might have converged, had $N < 1{,}000$ been used with MultiNest. MultiNest_IS' E/I is usually about -0.1 but the magnitude of its $\tilde{E}/\tilde{I}$ is usually significantly smaller than 0.1.

5.1.2 Significant Subcomputations of VoroInt

For $n = 10{,}000$, VoroInt executed the numbers of iterations of the loops shown in Table 5.1.2-1 (below). The iterations of the while loop are more significant than those of the for loop because the former involves more sites than the latter. However, there are fewer of the former than the latter.

Loop	Average	Maximum
While	3	5
For	7	13

Table 5.1.2-1: The numbers of iterations of the loops executed by VoroInt for $n = 10{,}000$

5.2 Gaussian Function

In this section, for $d = 2..6$, $G(\mathbf{x}; d)$ is integrated by VoroInt with $\mathrm{tol}_{\mathrm{rel}} = 0.1$, $n_{\min} = 50 \times 2^d$ (that is, 50 samples per quadrant, octant, …), and $n_{\max} = 10{,}000$. The function is then integrated by the other methods using the number of samples used by VoroInt. As d increases, $I_G(d)$ decreases but V_D remains constant, so the difficulty of the detection and, therefore, integration of G increases. With MultiNest, $N = 10$ is used so that its working set may collapse around G after few evaluations.

5.2.1 Accuracy

Figures 5.2.1-1..5 (below), which are plots of accuracy versus n, show the points that correspond to the samples used by the methods to integrate G for $d = 2..6$. For $d = 4..5$, MultiNest's working set collapses at the value of n at which its output terminates, hence the correction to its E/I is insignificant but shown adjacent to the last point of its E/I. The points of MultiNest's $\tilde{E}/\tilde{I}$ that correspond to the corrections are about 0. Notice the following trends as d increases:

- The gap between VoroInt's E/I and $\tilde{E}/\tilde{I}$ at the beginning of their smooth phase widens.
- VEGAS' E/I and $\tilde{E}/\tilde{I}$ become and remain about 0 at approximately $n = 2{,}000$, which corresponds to 5 iterations with 400 samples each.
- The variances of MISER's E/I and $\tilde{E}/\tilde{I}$ over n after the first stratification

dramatically increase.

- PLAIN's E/I and $\tilde{E}/\tilde{I}$ degrade as expected as the difficulty of the integration of G increases.
- MultiNest's E/I alternates between indications of under- and overestimation of I_G (with the possibility of correction for $d = 2$).
- MultiNest_IS' E/I also degrades as expected and its $\tilde{E}/\tilde{I}$ becomes more unreliable.
- Nest's E/I tends toward identically -1 and, therefore, its $\tilde{E}/\tilde{I}$, which always rapidly decays, becomes more misleading.

Because VoroInt's E/I is typically 0.1 at the beginning of its smooth phase, the widening gap indicates a bias in VoroInt's $\tilde{E}/\tilde{I}$ for G that worsens as d increases. The bias is probably positive because it is caused by the overestimation of the integrals over the regions in which G is mostly concave up (which become more numerous as d increases). VEGAS is able to adapt so well using so few samples because G is separable. Although the variances dramatically increase, MISER's E/I and $\tilde{E}/\tilde{I}$ are typically better than those of PLAIN. For $d = 2$, MultiNest's E/I does not appreciably increase by $n = 200$ despite the detection of G evidenced by MultiNest_IS' E/I. Nest does not detect G for $d = 4..6$. VoroInt errs at about $n = 3{,}000$ because of a problem in Qhull that will be worked around in future versions of VoroInt.

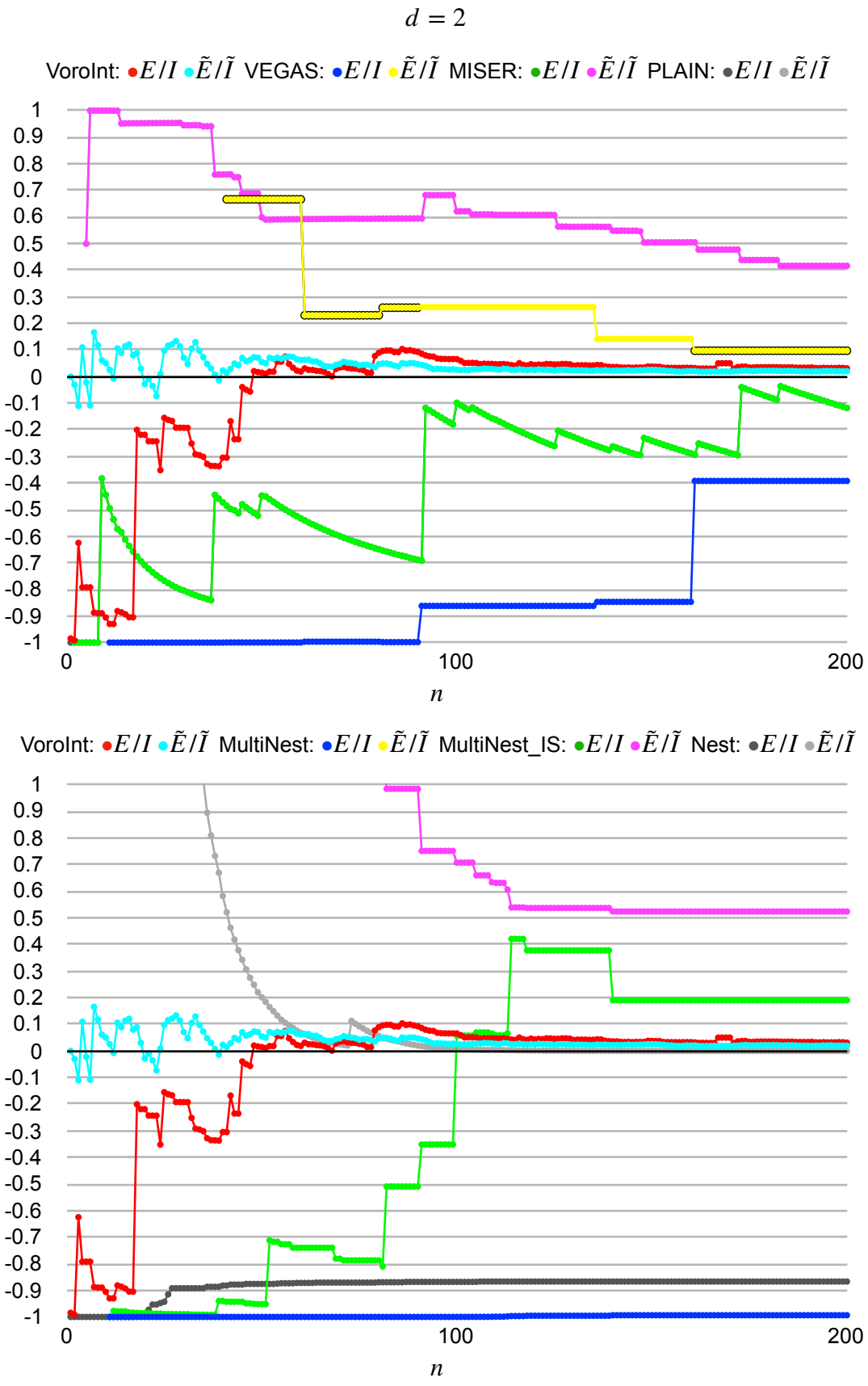


Figure 5.2.1-1: Accuracy versus n

$d = 3$

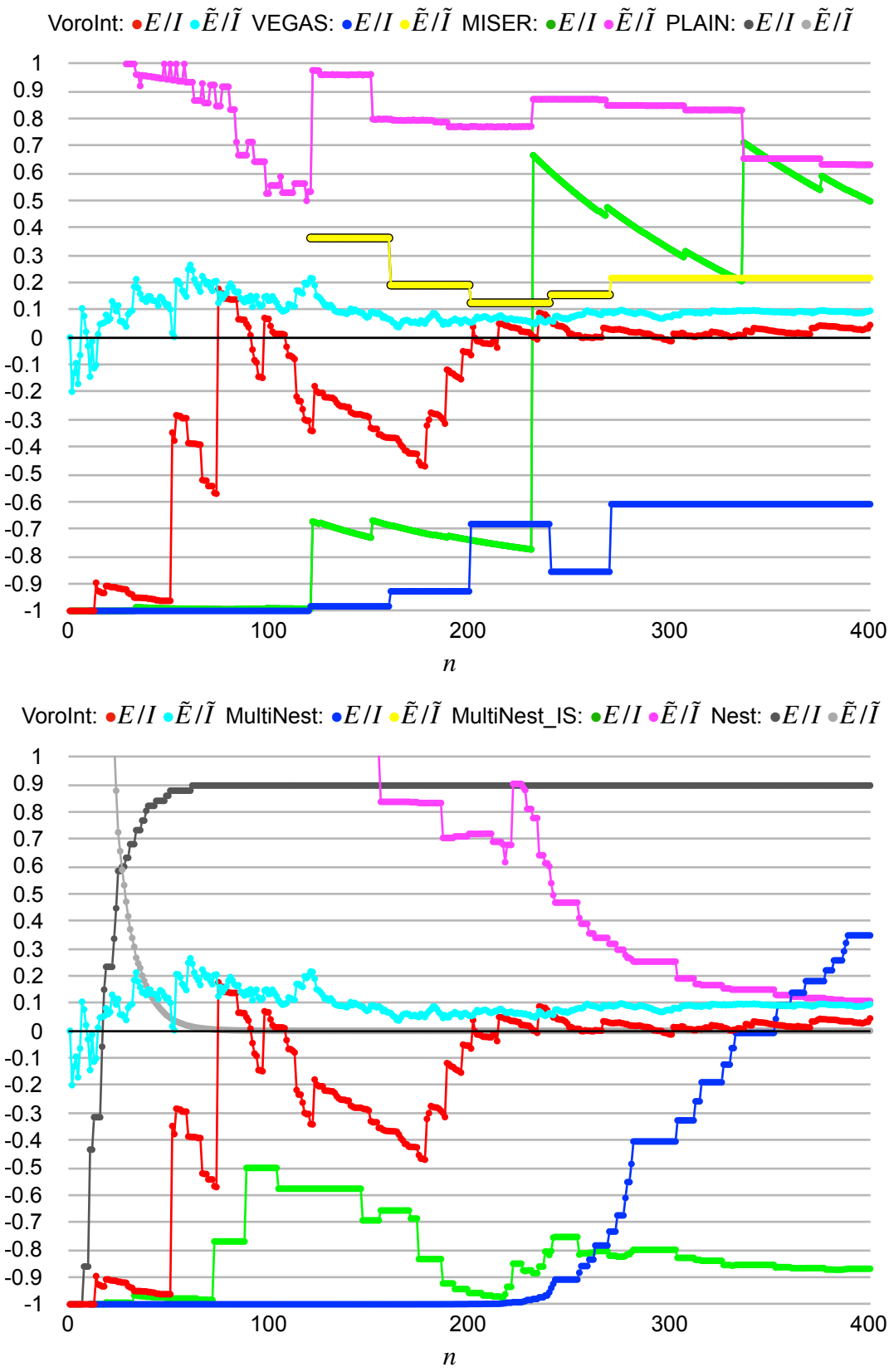


Figure 5.2.1-2: Accuracy versus n

$d = 4$

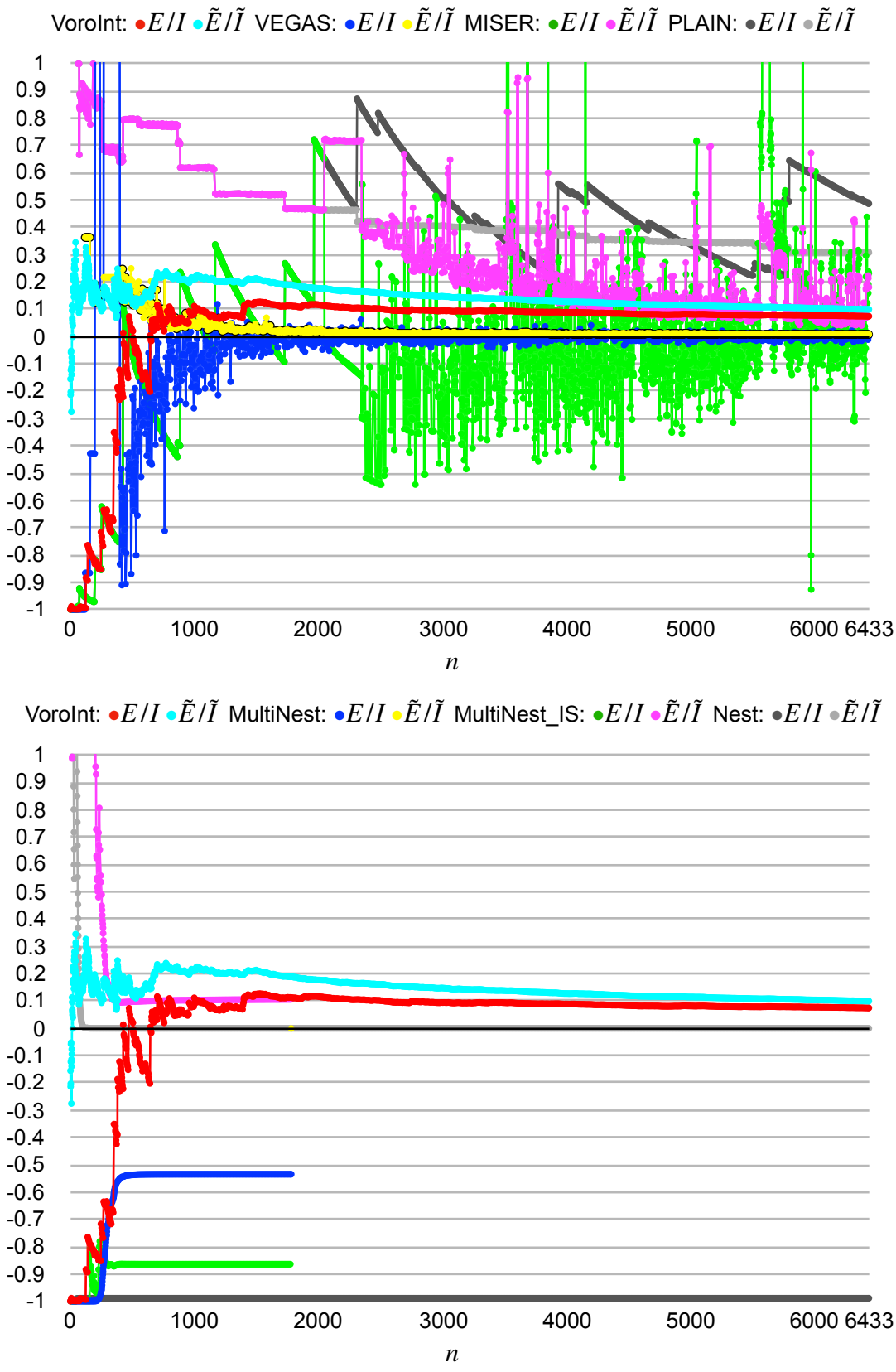


Figure 5.2.1-3: Accuracy versus n

$d = 5$

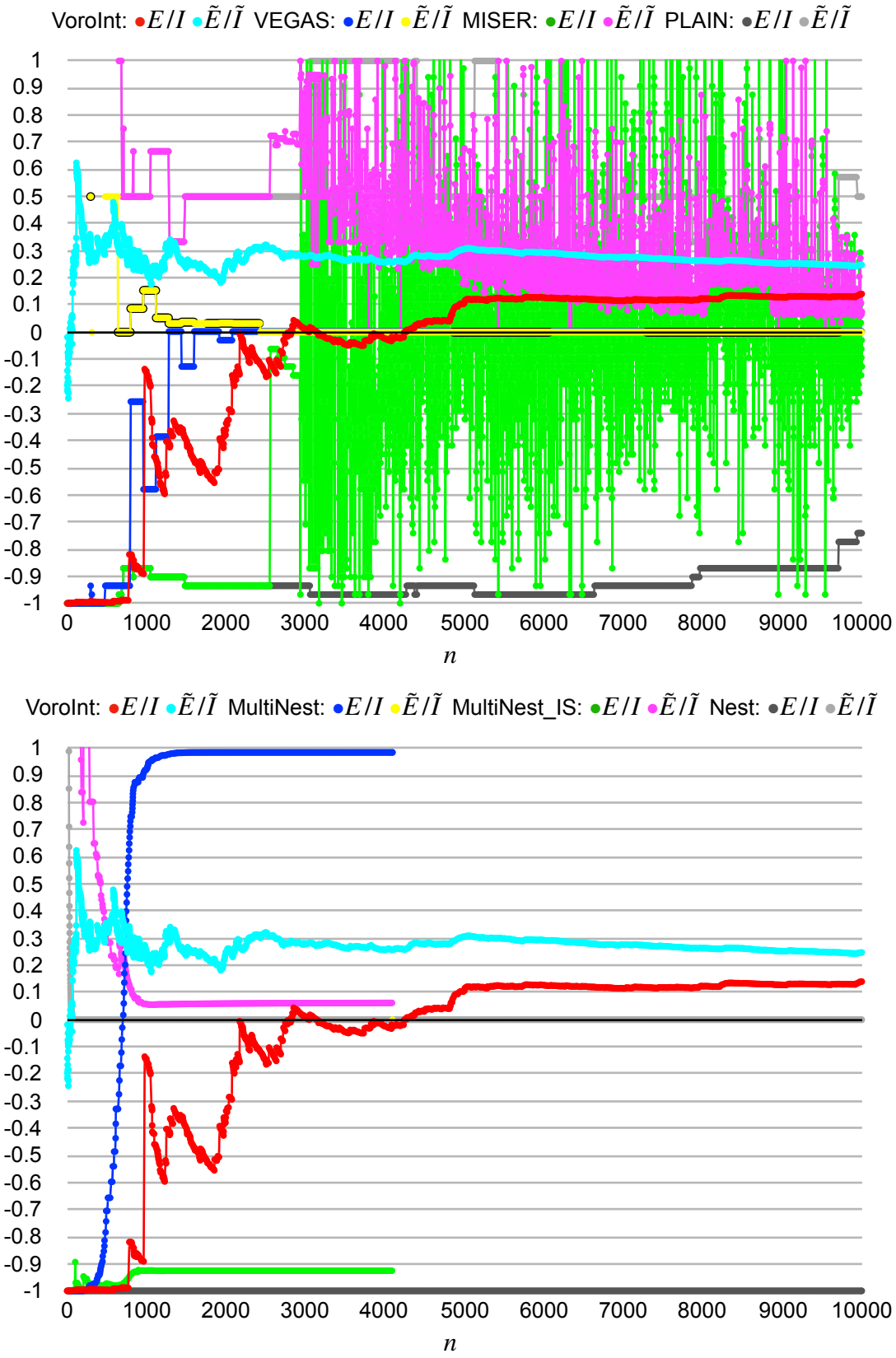
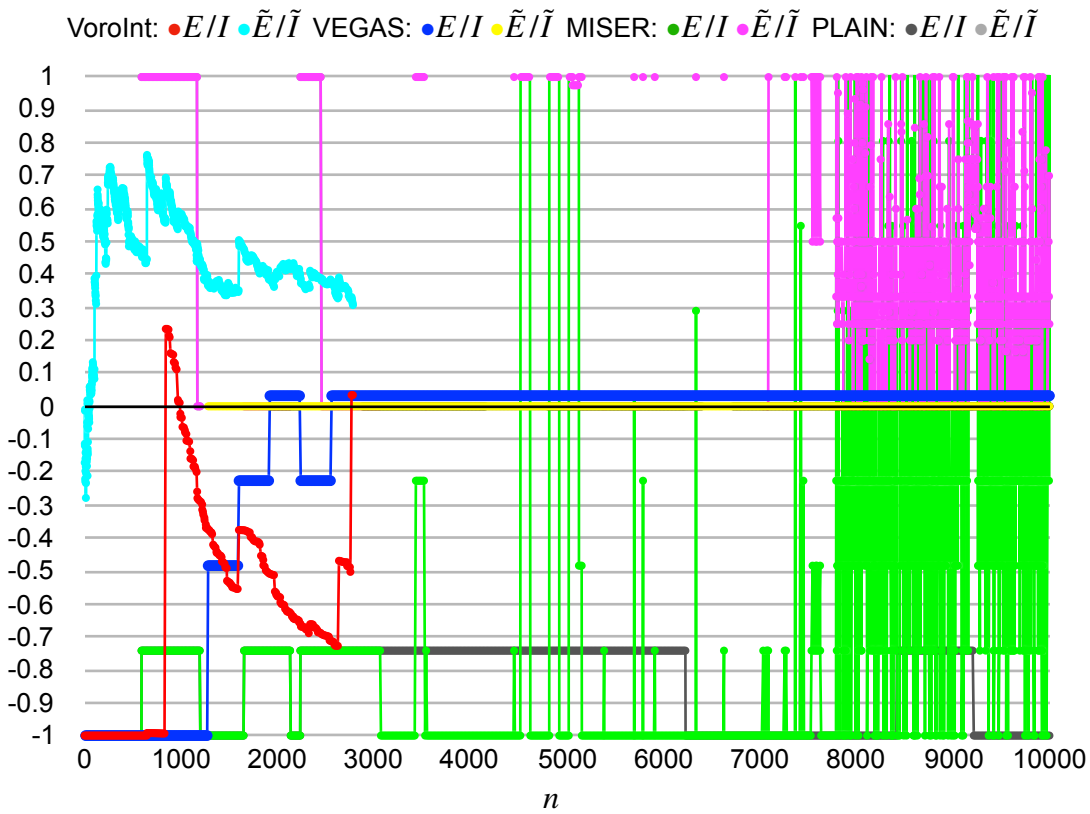


Figure 5.2.1-4: Accuracy versus n

$d = 6$



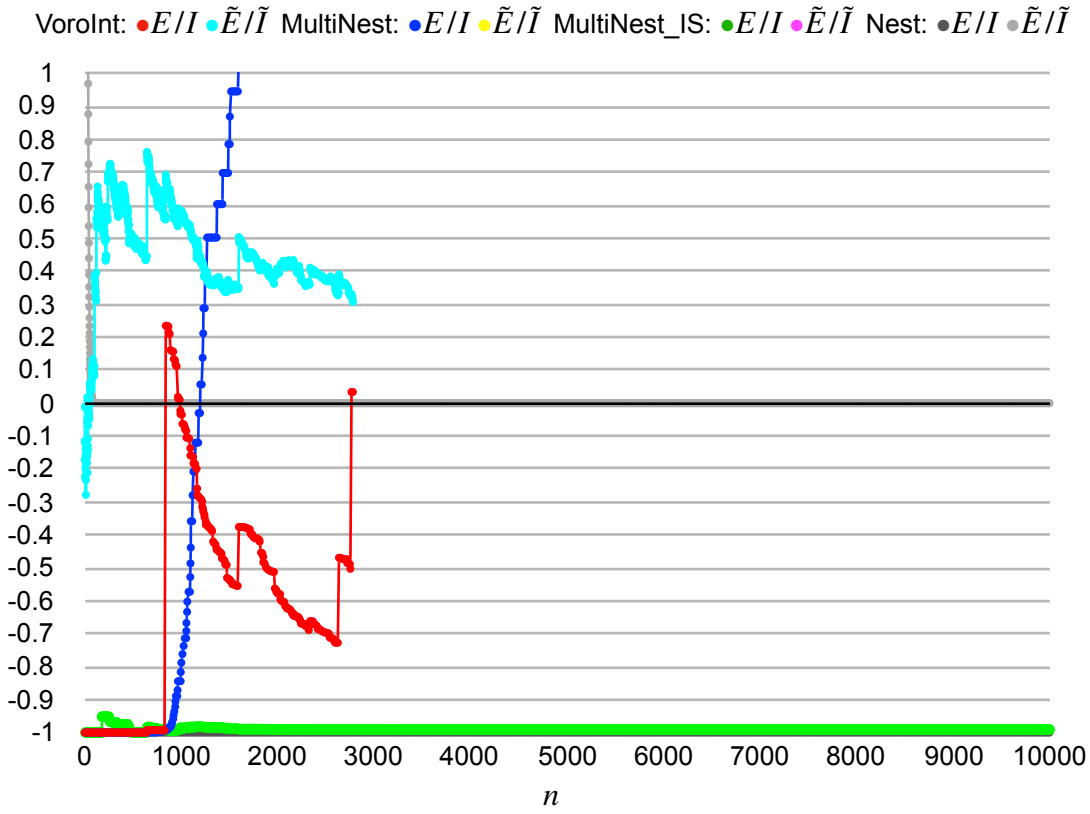


Figure 5.2.1-5: Accuracy versus n

5.2.2 Significant Subcomputations of VoroInt

For $d = 2..6$, VoroInt executed the following numbers of iterations of the while loop:

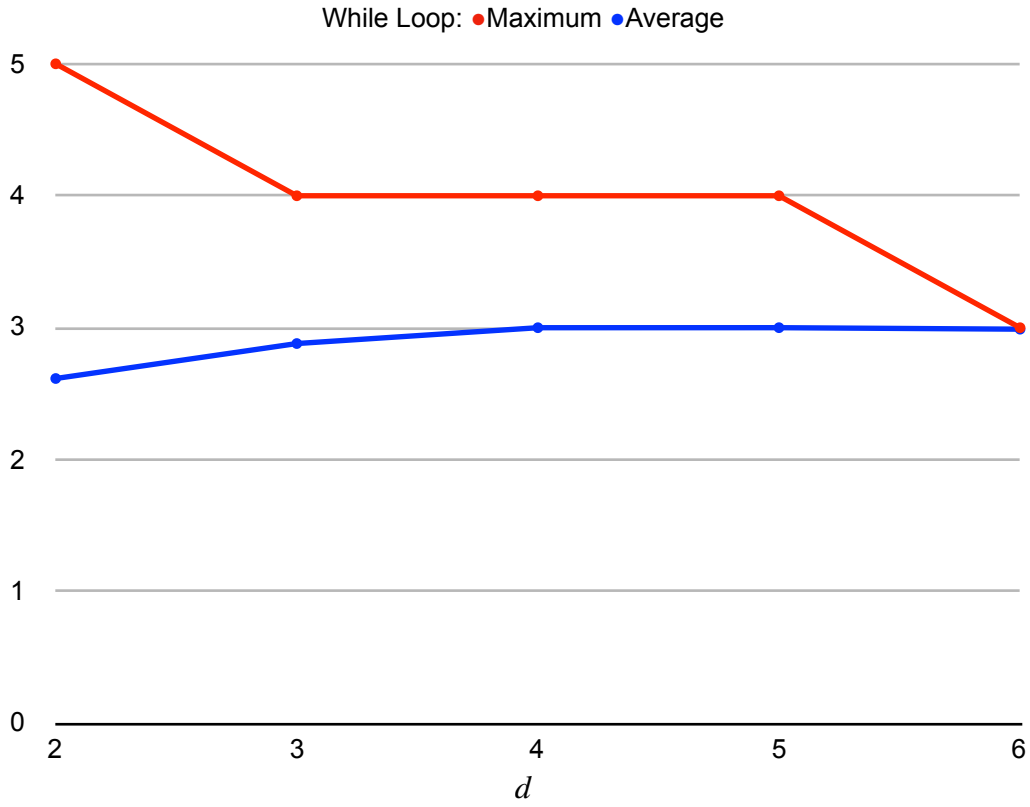


Figure 5.2.2-1: The numbers of iterations of the while loop executed by VoroInt

The averages for $d = 2..3$ could be lower than those for $d = 4..6$ because for the former the degrees of freedom are fewer than for the latter, so for the former an average neighborhood of the new site is more compact than that for the latter.

For $d = 2..6$, VoroInt executed the following numbers of iterations of the for loop:

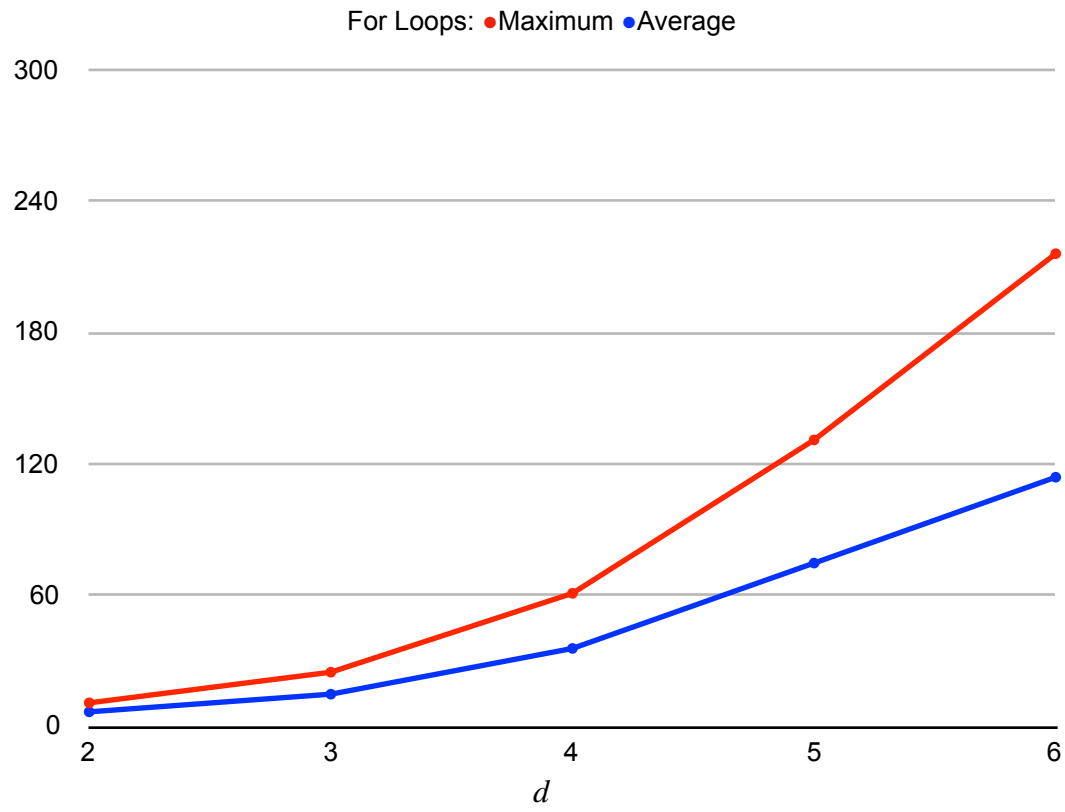


Figure 5.2.2-2: The numbers of iterations of the for loop executed by VoroInt

The numbers indicate the number of neighbors depends on d.

5.3 Gaussian Circle Function

In this section, for $d = 2..6$, $C(\mathbf{x})$ is integrated using $n = 1{,}000$ samples. As d increases, I_C remains constant, so the difficulty of the detection and integration of C is constant. With MultiNest, $N = 100$ is used so that it is able to cover the circle in ellipsoids.

5.3.1 Accuracy

Figures 5.3.1-1..5 (below), which are plots of accuracy versus n, show the points that correspond to the samples used by the methods to integrate C for $d = 2..6$. Because the difficulty of the detection and integration of C is low, the methods should behave well. Notice VEGAS' E/I and $\tilde{E}/\tilde{I}$ take longer to converge as d increases because C is inseparable.

$d = 2$

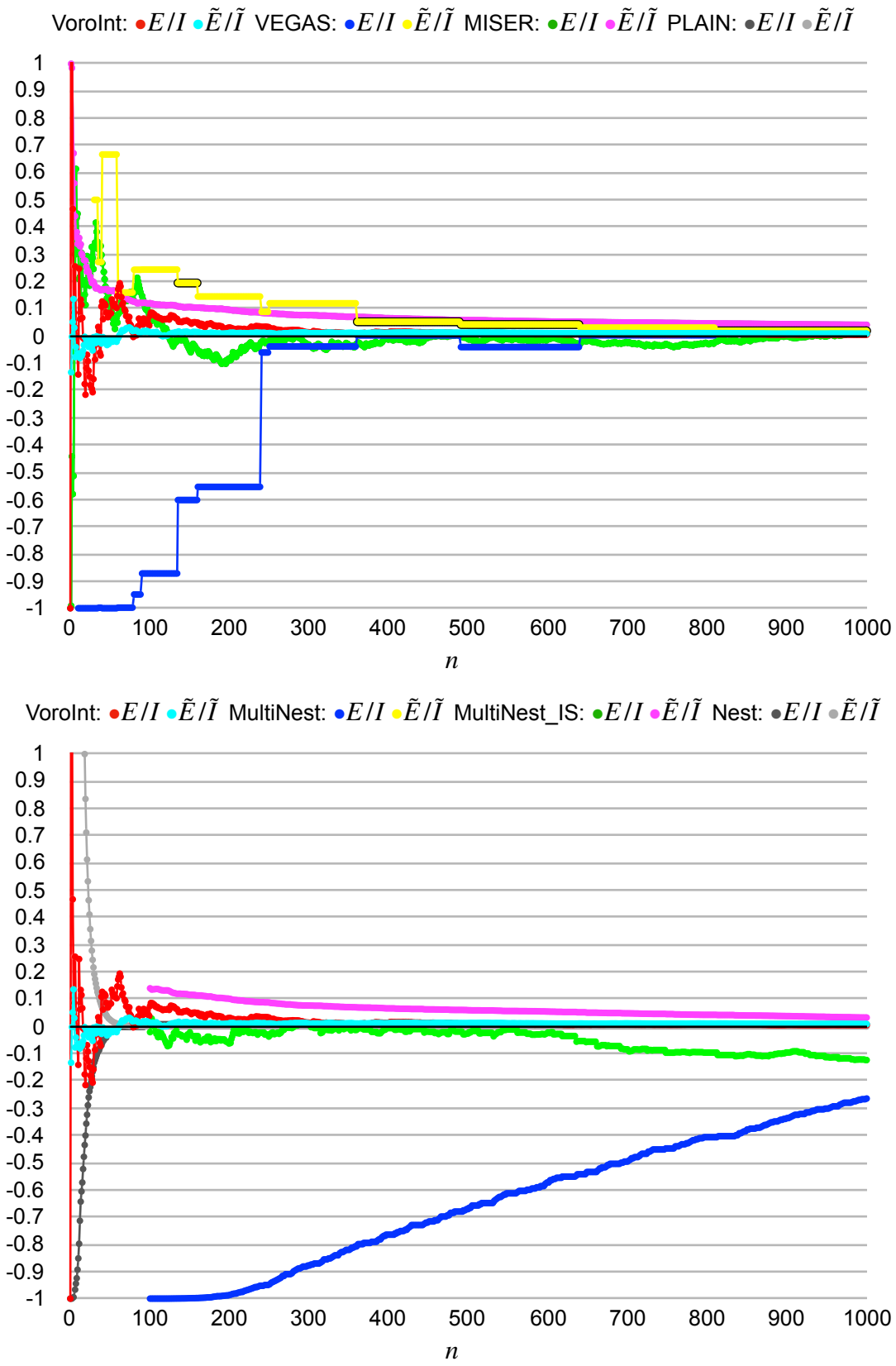
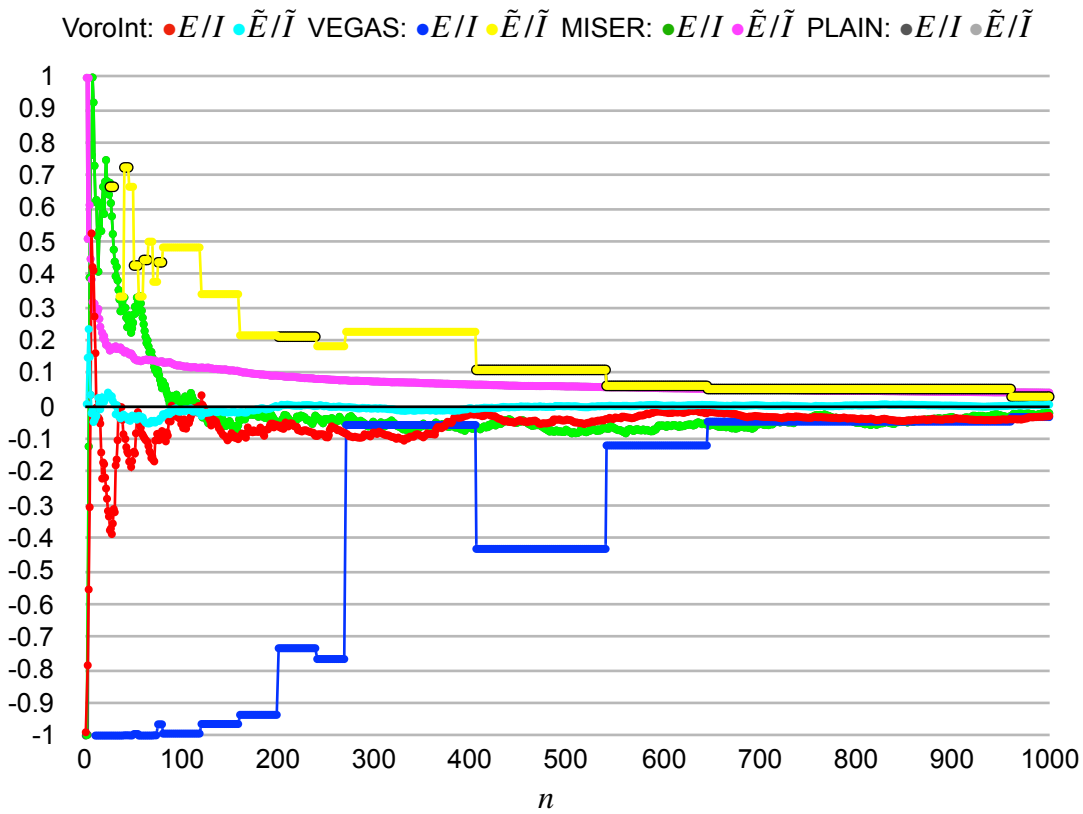


Figure 5.3.1-1: Accuracy versus n

$d = 3$



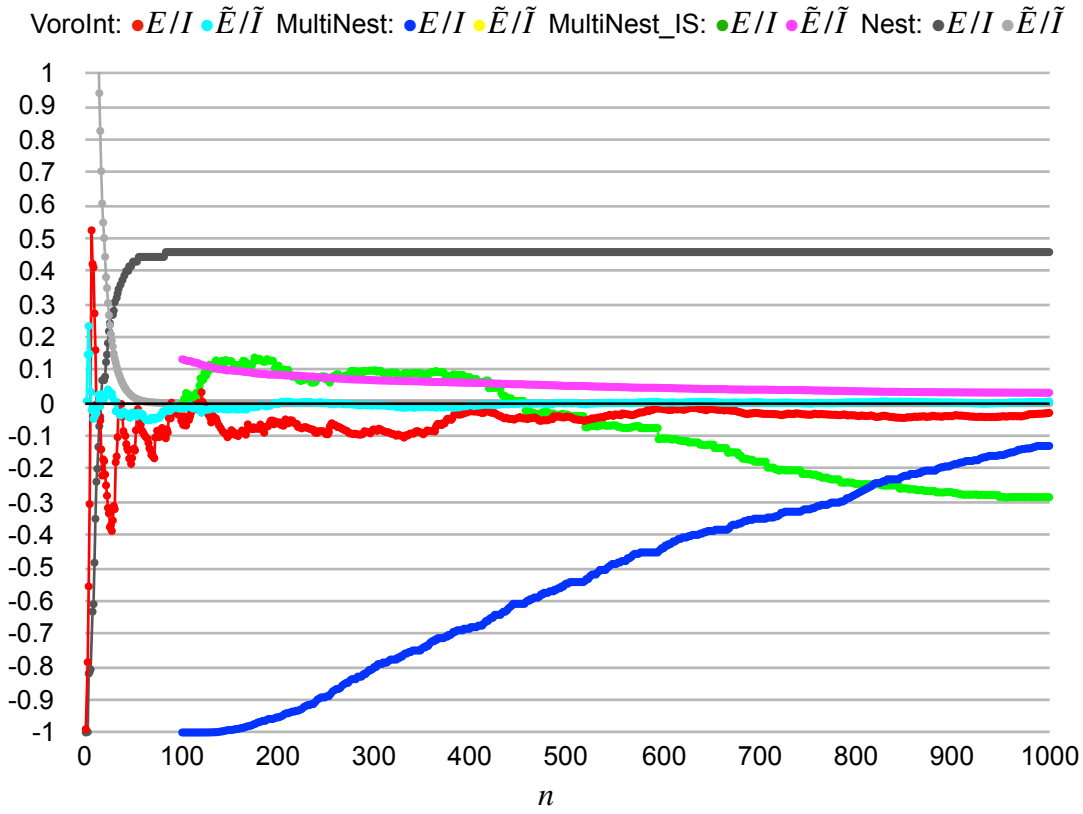


Figure 5.3.1-2: Accuracy versus n

$d = 4$

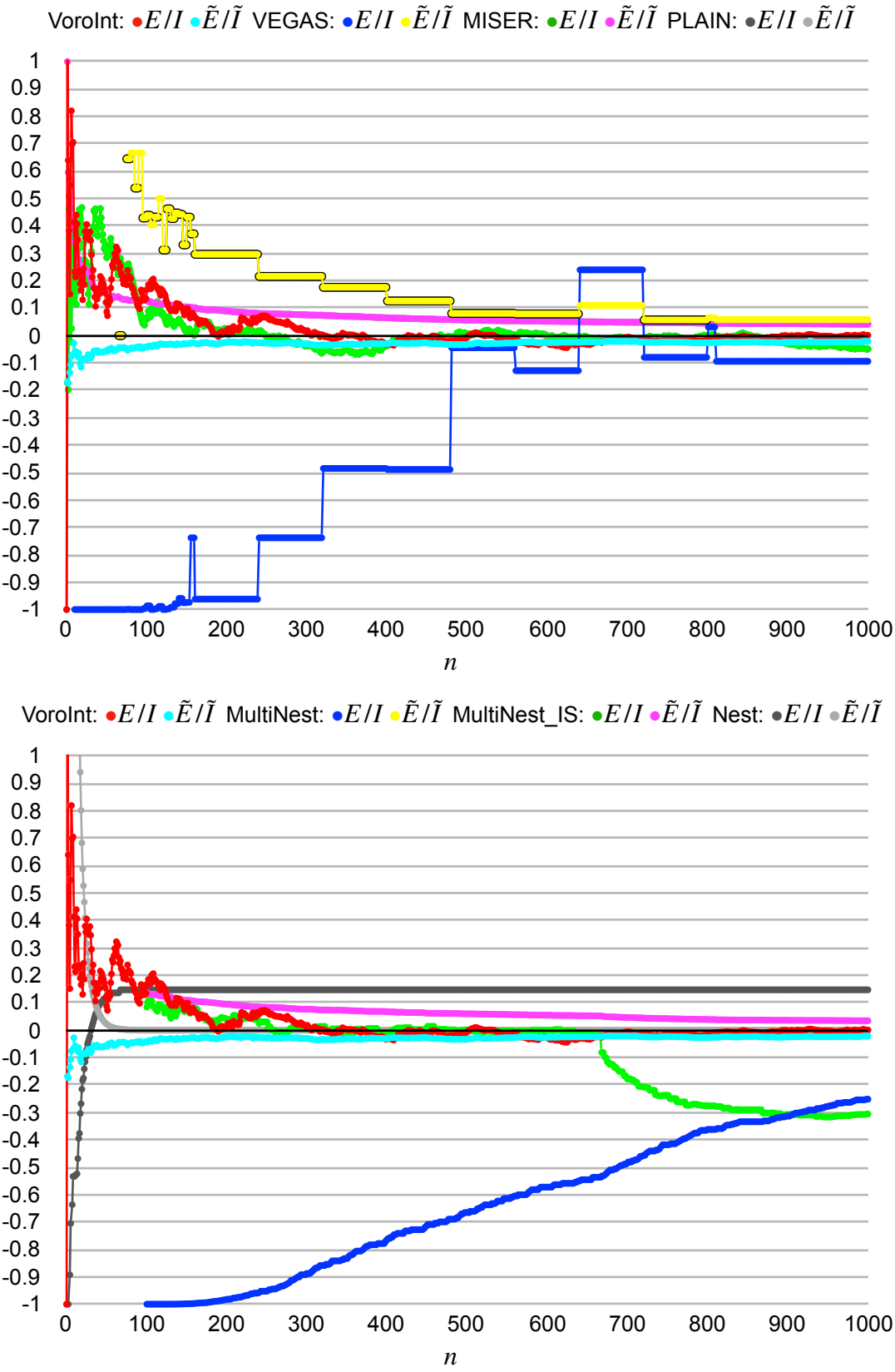


Figure 5.3.1-3: Accuracy versus n

$d = 5$

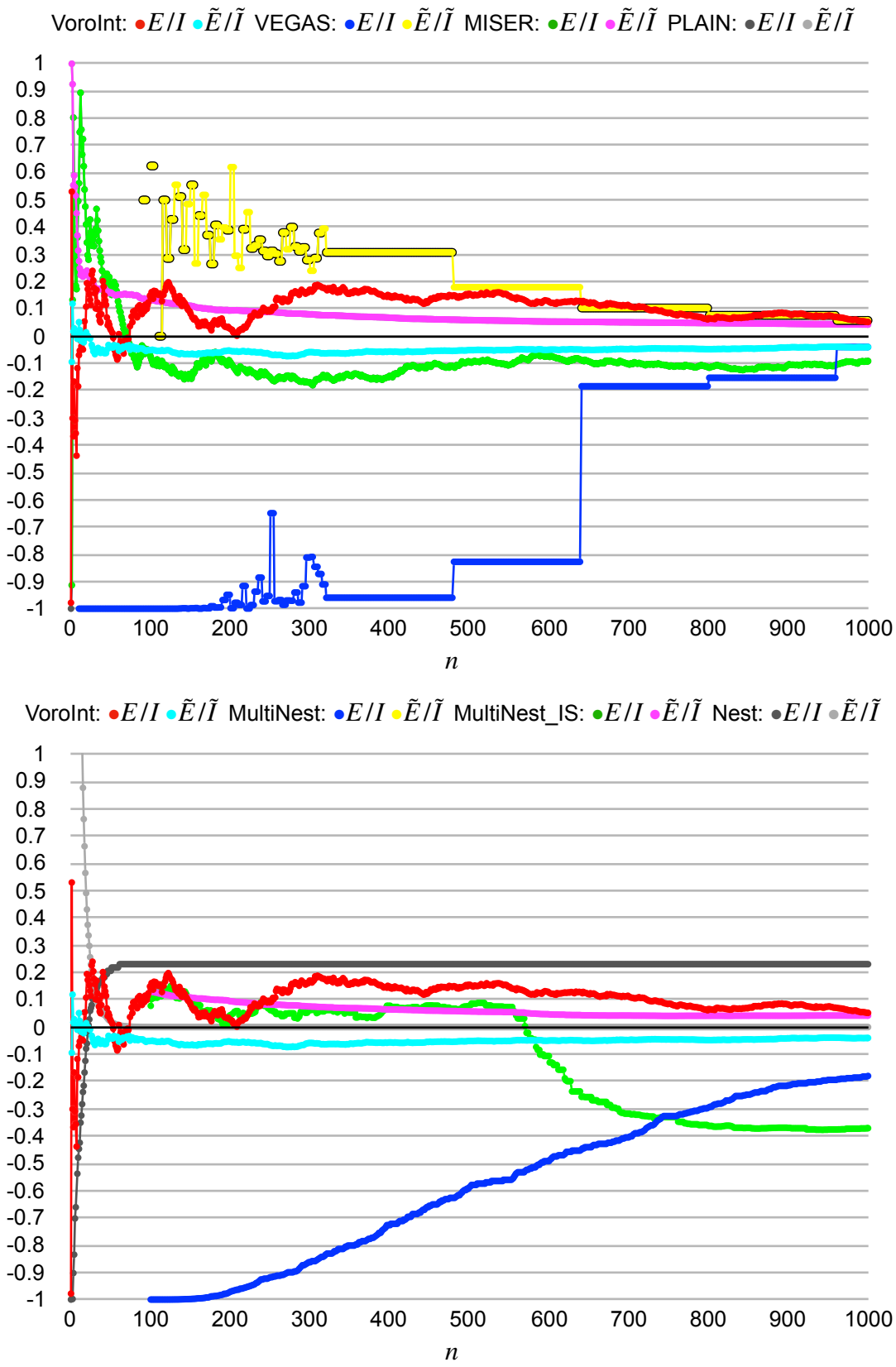


Figure 5.3.1-4: Accuracy versus n

$d = 6$

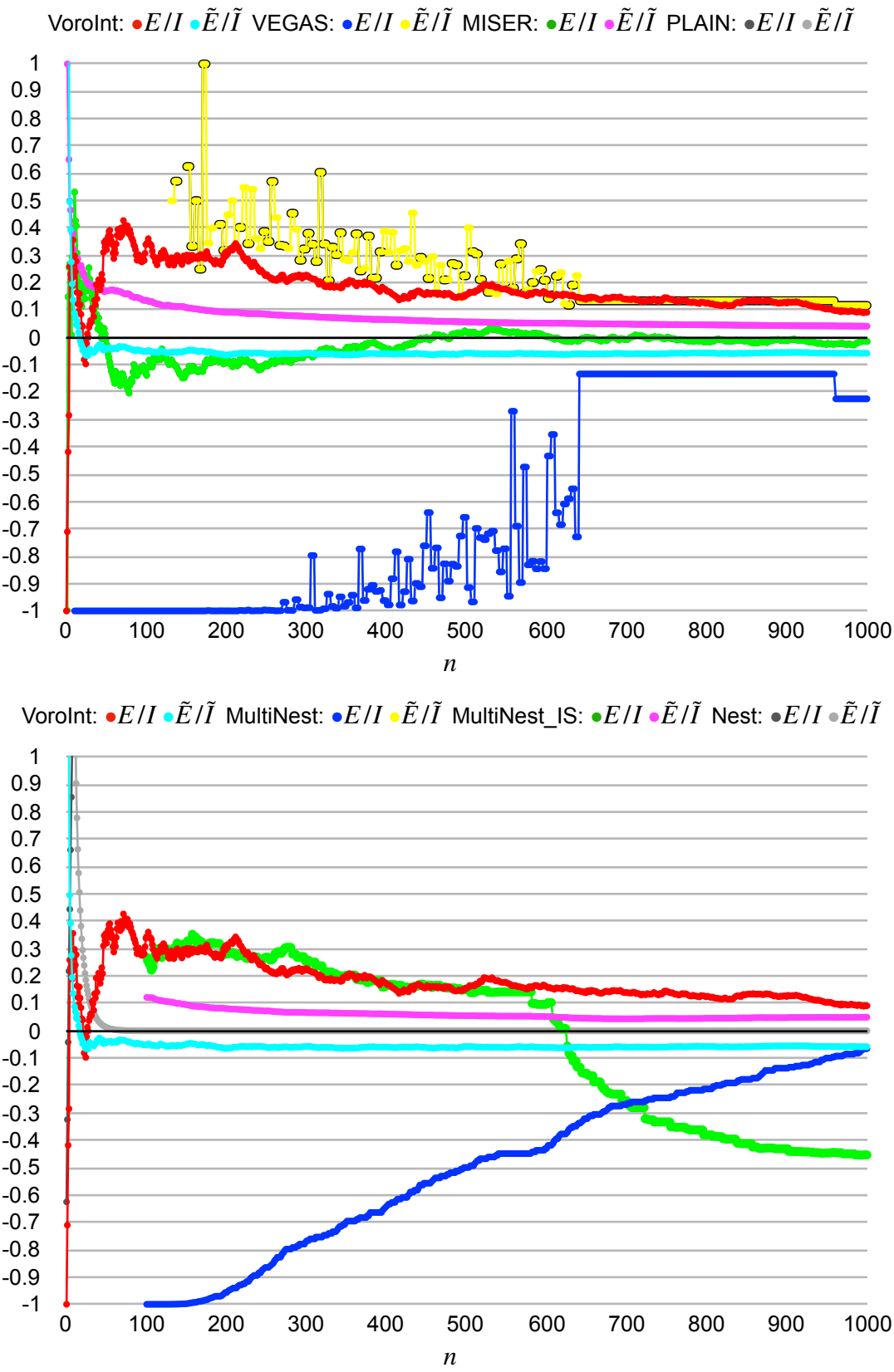


Figure 5.3.1-5: Accuracy versus n

As d increases, the inseparability of C is increasingly exaggerated and, consequently, VEGAS is increasingly unable to adapt. MISER (or PLAIN) behaves very well. VoroInt behaves well and its $\tilde{E}/\tilde{I}$, which is typically positive for the other integrands, is typically negative, which could be because many neighborhoods span the ridge. MultiNest behaves well and could converge to 0 for $n > 1{,}000$. MultiNest_IS behaves well until the behavior of its E/I abruptly changes, at which point the behavior of its $\tilde{E}/\tilde{I}$ becomes increasingly errant. Nest behaves ill for $d > 2$.

5.3.2 Significant Subcomputations of VoroInt

For $d = 2..6$, VoroInt executed the following numbers of iterations of the while loop:

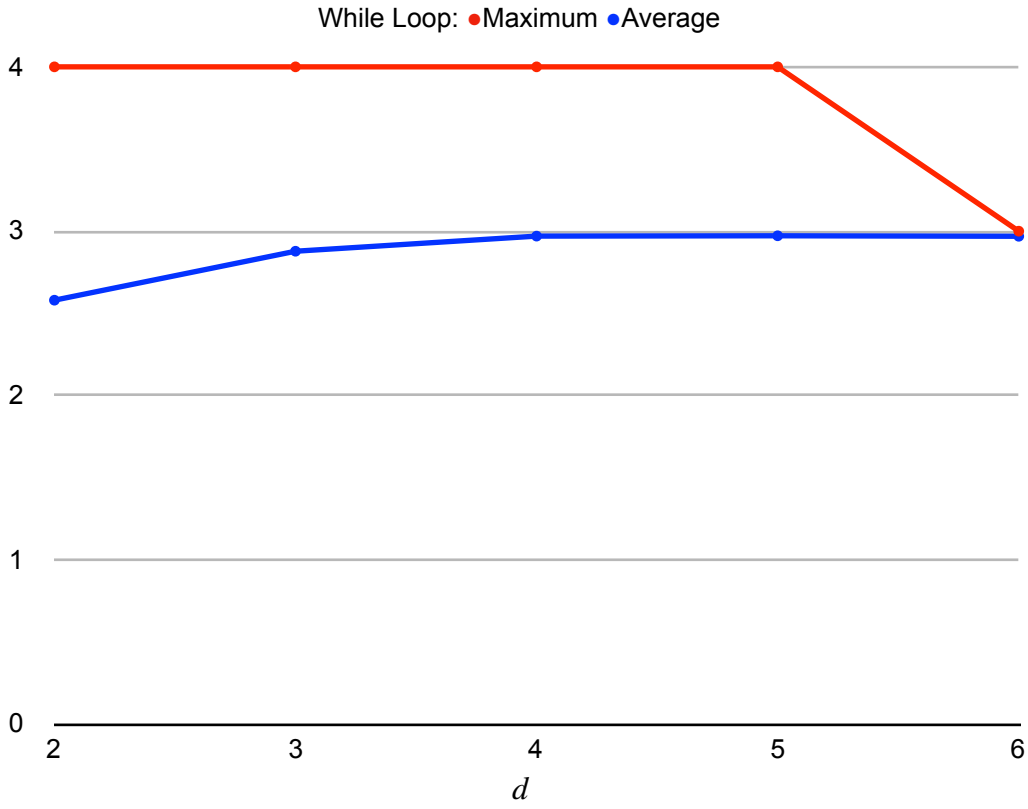


Figure 5.3.2-1: The numbers of iterations of the while loop executed by VoroInt

These numbers, as with those for G, indicate the control of the while loop does not depend on d. However, the number of sites involved in each iteration of the while loop depends on the number of neighbors of the site

being inserted into the Voronoi decomposition, which is precisely the number of iterations of the subsequent for loop. And, as the numbers of iterations of the for loop for G indicate, the number of neighbors depends on d.

For $d = 2..6$, VoroInt executed the following numbers of iterations of the for loop:

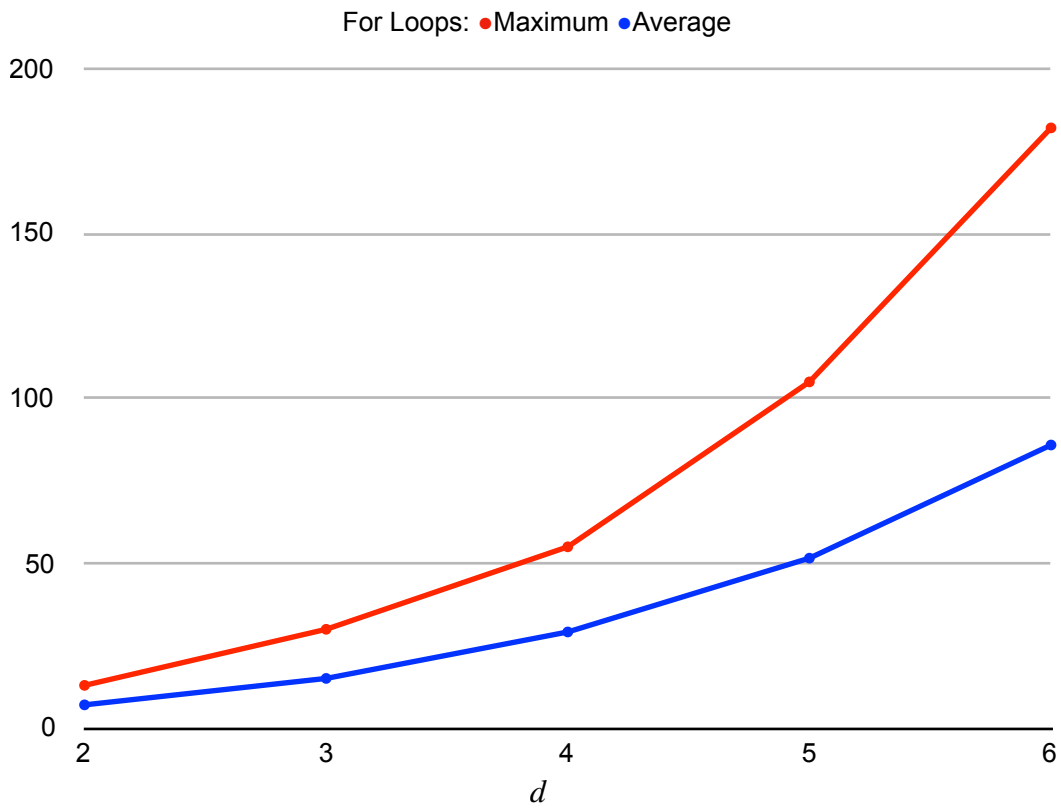


Figure 5.3.2-2: The numbers of iterations of the for loop executed by VoroInt

Again, the numbers indicate the number of neighbors depends on d.

5.4 Summary

In this section, the accuracy results are summarized. Also, the significant subcomputations of VoroInt are used to estimate its computational complexity.

5.4.1 Accuracy

Table 5.4.1 (below) shows, for each method, the values of E/I and $\tilde{E}/\tilde{I}$ that

correspond to select points from the plots of accuracy versus n, grouped by f, d, and n. For each group, the worst (that is, highest) value of $|E/I|$ is highlighted in red and the best (that is, lowest) value of $|E/I|$ is highlighted in green. The values of $\tilde{E}/\tilde{I}$ that ill approximate their companion values of E/I are highlighted in yellow to signal their unreliability. $\tilde{E}/\tilde{I}$ ill approximates its companion E/I if and only if $0.5 < \frac{|E/I|}{\tilde{E}/\tilde{I}} < 2.0$ (that is, $\tilde{E}/\tilde{I}$ is within a factor of 2 of $|E/I|$). For VEGAS, the values that are derived from inconsistent results are shown in magenta. For MultiNest, when $\tilde{E}/\tilde{I}$ exists, the output terminates in fewer than n samples. For VoroInt, the values of $\tilde{E}/\tilde{I}$ that are incorrectly signed are shown in blue.

PLAIN is the best for some of the groups for which f includes C, which is less difficult to integrate than G and dominates $G + C$. VEGAS is the worst for $n = 100$ but the best for G (which is separable) and $d \geq 4$, albeit with values that are derived from inconsistent results, and, surprisingly, for C (which is inseparable) and $d = 5$. Unsurprisingly, given VEGAS' performance on G, a modified version of VEGAS has been used in cosmology ([32]). MISER is only the best when it is equal to PLAIN (which is before its first stratification). Nest is the worst most often and its $\tilde{E}/\tilde{I}$ is always unreliable. MultiNest is the worst second-most often and its $\tilde{E}/\tilde{I}$ is always (when it exists) unreliable. MultiNest_IS is the worst third-most often and its $\tilde{E}/\tilde{I}$ is always unreliable. VoroInt is the best most often and for almost half of the groups and its $\tilde{E}/\tilde{I}$ is the most reliable, albeit with values that are incorrectly signed for C and $d \geq 3$. However, for G and $d = 6$, VoroInt's run erred at $n = 2{,}785$.

f	d	n	*	PLAIN	VEGAS	MISER	Nest	MultiNest		VoroInt
									IS	
$G+C$	2	100	E/I	0.066	-0.684	0.066	0.173			0.038
			$\tilde{E}/\tilde{I}$	0.114	0.328	0.114	0.000			0.004
		1,000	E/I	0.004	0.021	0.004	0.173			0.009
			$\tilde{E}/\tilde{I}$	0.038	0.019	0.038	0.000			0.008
		10,000	E/I	-0.002	0.002	-0.003	0.173	-0.178	-0.068	0.001
			$\tilde{E}/\tilde{I}$	0.012	0.002	0.013	0.000		0.010	0.001
G	2	200	E/I	-0.118	-0.392	-0.118	-0.867	-0.994	0.191	0.031
			$\tilde{E}/\tilde{I}$	0.417	0.097	0.417	0.000		0.524	0.020
	3	400	E/I	0.498	-0.610	0.498	0.897	0.349	-0.870	0.046
			$\tilde{E}/\tilde{I}$	0.631	0.217	0.631	0.000		0.110	0.098
	4	6,433	E/I	0.487	0.001	0.232	-0.989	-0.535	-0.864	0.075
			$\tilde{E}/\tilde{I}$	0.308	0.008	0.181	0.000	-0.000	0.106	0.100
	5	10,000	E/I	-0.741	0.002	-0.127	-1.000	0.986	-0.926	0.141
			$\tilde{E}/\tilde{I}$	0.500	0.000	0.074	0.000	0.000	0.062	0.248
	6	10,000	E/I	-1.000	0.032	0.032	-1.000	2.182	-0.991	0.034
			$\tilde{E}/\tilde{I}$	0.000	0.000	0.250	0.000		3.3E+07	0.305
C	2	1,000	E/I	0.007	0.009	0.007	0.008	-0.265	-0.124	0.005
			$\tilde{E}/\tilde{I}$	0.041	0.019	0.041	0.000		0.032	0.009
	3	1,000	E/I	-0.021	-0.033	-0.021	0.459	-0.130	-0.285	-0.029
			$\tilde{E}/\tilde{I}$	0.041	0.029	0.041	0.000		0.032	0.005
	4	1,000	E/I	-0.049	-0.094	-0.049	0.148	-0.252	-0.306	0.001
			$\tilde{E}/\tilde{I}$	0.042	0.058	0.042	0.000		0.035	-0.022
	5	1,000	E/I	-0.091	-0.040	-0.091	0.231	-0.179	-0.371	0.052
			$\tilde{E}/\tilde{I}$	0.043	0.058	0.043	0.000		0.042	-0.040
	6	1,000	E/I	-0.015	-0.225	-0.015	4.300	-0.065	-0.454	0.092
			$\tilde{E}/\tilde{I}$	0.041	0.117	0.041	0.000		0.050	-0.058

Table 5.4.1-1: Accuracy versus n

5.4.2 Significant Subcomputations of VoroInt

As d increases, VoroInt asymptotically averages 3 iterations of the while loop. VoroInt averages a number of iterations of the for loop that depends on d and f. For example:

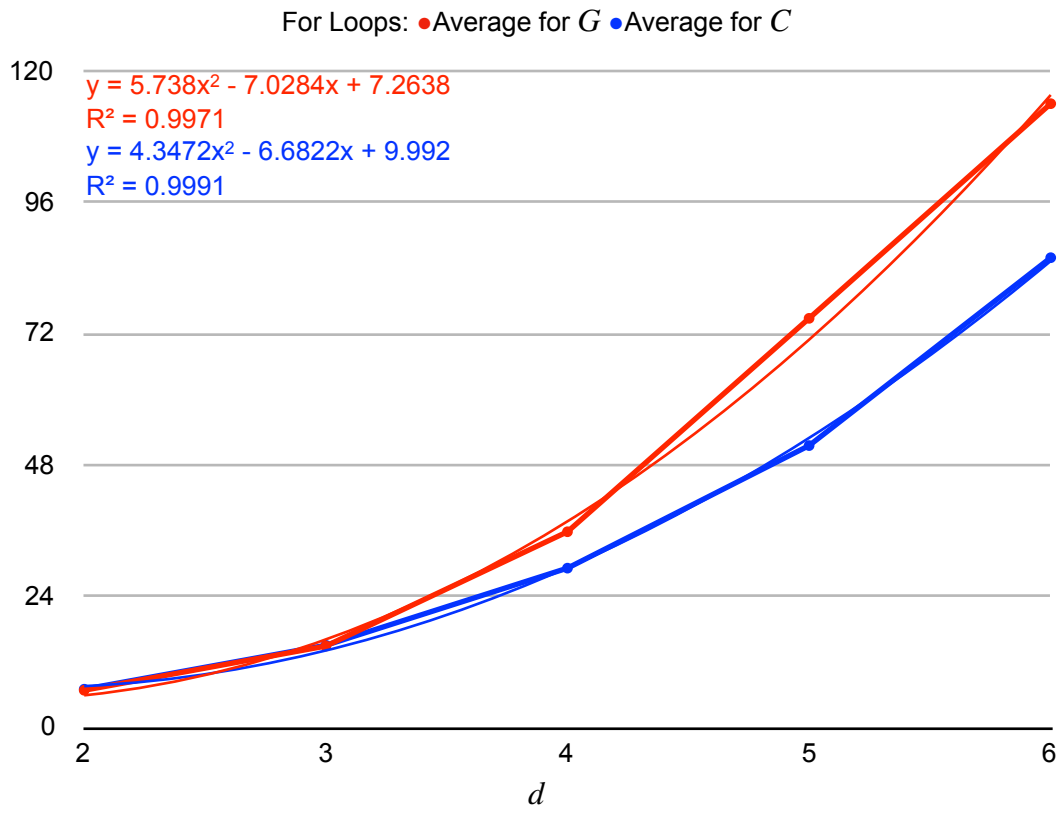


Figure 5.4.2-1: The dependence on d and f of the average number of iterations of the for loop executed by VoroInt

Seemingly, the number of for loops is quadratic in d with the integrand-specific coefficients c_0, c_1, and c_2.

Substituting $O(3) = O(1)$ and $O(c_2 d^2 + c_1 d + c_0) = O(d^2)$ for l and c in $O(n(lc^{2\lceil d/2 \rceil} + c(c+2d)^{\lceil d/2 \rceil}))$, which is the complexity of VoroInt from Section 4.1.4, yields $O(n(d^{2(2\lceil d/2 \rceil)} + d^2(d^2+2d)^{\lceil d/2 \rceil})) = O(n(d^{4\lceil d/2 \rceil} + d^{2+2\lceil d/2 \rceil}))$ because $O(d^2 + 2d) = O(d^2)$. For all $d > 2$, $d^{4\lceil d/2 \rceil}$ dominates $d^{2+2\lceil d/2 \rceil}$. Therefore, the complexity of VoroInt is $O(n d^{4\lceil d/2 \rceil})$, which bodes ill for the

applicability of VoroInt for $d \geq 9$ for which on the order of 10^{20} operations are required per insertion if the upper bound is tight. However, if the factor of c^2 is more like a factor of c, which corresponds to the case in which all of the c sites are mutual neighbors, and the c for loops can be concurrently executed, then the complexity is $\Omega(n(lc^{2\lceil d/2 \rceil} + c(c+2d)^{\lceil d/2 \rceil})) = \Omega(n(lc^{\lceil d/2 \rceil} + (c+2d)^{\lceil d/2 \rceil}))$. Substituting in the lower bound yields $\Omega(n(d^{2\lceil d/2 \rceil} + d^{2\lceil d/2 \rceil})) = \Omega(nd^{2\lceil d/2 \rceil})$, which bodes better for $d \geq 9$ for which on the order of 10^{10} operations are required per insertion per processor.

6 Conclusion

Using the terminology and the demonstration function introduced in Chapter 1, PLAIN, VEGAS, and MISER, which are the standard Monte Carlo Integration methods, were described in Chapter 2, Nest and MultiNest, which are nested sampling (integration) methods, were described in Chapter 3, and VoroInt, which is a novel numerical integration method based on Voronoi Integration, was described in Chapter 4. The results of the methods on the demonstration function and its components, which are a Gaussian function and a Gaussian Circle function, were compared in Chapter 5.

The results indicate that, on multidimensional integrands that are relevant to cosmology, the evaluation efficiency of VoroInt is higher than or comparable to that of the other methods, except on a 4^+-dimensional Gaussian on which VEGAS excels. Furthermore, the error estimate of VoroInt is reliable so it can be used to terminate the method with necessary confidence in the result of the method but without unnecessary evaluations of the integrand, which are expensive in cosmological applications.

The main strength and weakness for each of the methods is:

Method	Strength	Weakness
PLAIN	Simplicity	Difficulty of detection and resolution of small features of integrands
VEGAS	Efficacy on separable integrands, especially ones with a single small feature	Inefficacy on inseparable integrands, especially ones with a single large feature or multiple features
MISER	Optimality of allocation of evaluations among strata	Immutability of evaluation budget, which is set at runtime
Nest	Applicability to integrands with ultra-high dimensionality	Lack of prescription for sampling from level sets of integrands
MultiNest	Efficacy on integrands with multiple peaks or low-dimensional ridges	Sensitivity to choice of value of N, which is inherited from Nest
MultiNest_IS	Use of all evaluations by MultiNest	Dependence on MultiNest
VoroInt	Efficiency of evaluation	Cost of computation

Table 6-1: The main strength and weakness for each of the methods

PLAIN's strength causes its weakness because its simplicity precludes adaptation. VEGAS' and MISER's strengths and weaknesses are inherent to their design. MISER's weakness limits its use to cases in which the evaluation budget, rather than the accuracy target, is set at runtime. Nest's strength cannot be realized with a reasonable acceptance ratio unless its weakness is addressed in some manner, for example, by way of ellipsoidal nested sampling as in MultiNest. MultiNest's weakness, which is an auxiliary weakness of Nest, also affects MultiNest_IS because of MultiNest_IS' dependence on MultiNest. VoroInt's weakness is the source of its strength because VoroInt assumes evaluations are expensive and, so, maximizes their utility at considerable cost (which depends on the dimensionality).

The auxiliary strengths of VoroInt are:

- Ability to use hints with (or without) integrand (See Section 4.1.)
- Simplicity of parameterization (See Section 4.1.2.)
- Applicability to any integrand regardless of feature multiplicity (See Section 5.1.), feature size (See Section 5.2.), degeneracy (See Sections 5.1 and 5.3.), or separability (See Section 5.2 for separable and Sections 5.1 and 5.3 for inseparable.)
- Reliability of error estimate (See Section 5.4.1.)

The auxiliary weaknesses of VoroInt are:

- Dependence on Qhull (or other computational geometry package) (See Section 4.1.)
- Necessity of a guard against premature termination (See Sections 4.1 and 4.1.2.)
- Evidence of bias (See Section 5.2.)

Some ideas on how to address the weaknesses of and to extend VoroInt are in the following section. To aid comprehension of the discussion of the ideas, a summary of the operation of the method is:

VoroInt uses the Voronoi decomposition of the domain of integration about samples of the integrand to determine, for each sample, its natural neighbors. The neighbors of a sample are the samples in the Voronoi regions adjacent to its region and are determined using Qhull. The weight in a Riemann sum of a sample is the (hyper-)area of its Voronoi region, which is defined by its neighbors and the domain and is computed using Qhull. The neighbors of a sample also define the estimate of the error in its term in the sum and the priority of the sampling of its region. For a sample, the error estimate is its weight times the average deviation of its value from those of its neighbors and the sampling priority is its weight times the absolute maximal deviation of its value from those of its neighbors. The accurate error estimate enables timely termination. The effective sampling priority enables efficient error reduction. The sampling of a region is facilitated by its bounding box, which is constructed and stored while its weight is computed. Crucially, a sample is inserted into the implicit global Voronoi decomposition using an explicit local Voronoi decomposition, which involves few samples. The global Voronoi decomposition is implicit because it is represented by the adjacency information (that is, the natural neighbors for each sample) rather than the Voronoi vertices, which are too numerous to store when the dimensionality is high. The significant reduction in storage by way of the implicit representation of the global Voronoi decomposition would be unrealizable without a means to construct explicit local Voronoi decompositions (which are represented by Voronoi vertices) given the implicit representation. The while loop of the algorithm for inserting a sample into a Voronoi decomposition is such a means. Without the significant reduction in storage, the method would be unviable on ~10^{+}-dimensional integrands.

Because the cost of inserting a site into a Voronoi decomposition is considerable, the time-efficiency of VoroInt considerably increases as the cost of the evaluation of the integrand at a site approaches the cost of the insertion of a site. If the cost of an evaluation is less than the cost of an insertion, then VoroInt could run longer than other methods despite its

evaluation-efficiency. If the cost of an evaluation is greater than or equal to the cost of an insertion, then VoroInt should run shorter than other methods, especially when the alternative parallelization scheme described in Section 4.1.3 with an appropriate value of t is used. If t times the cost of an insertion is less than or equal to the cost of an evaluation, then the effective cost of an insertion is nothing. The parallelization of VoroInt is flexible but can be complicated because parallelism exists in both the insertion of a site into a Voronoi decomposition and the evaluation of an integrand. When $d \geq 5$ and the number of processors available for the insertion of a site is less than the value corresponding to d in Figure 5.4.2-1, the insertion cost will be substantially more than a second.

In cosmological model selection, the time-efficiency of the method used to compute the Bayesian evidence for each model is important, especially when there are many plausible models. However, when two models are vying for selection, the accuracy of the method and the reliability of its error estimate are more important because, to enable selection, the Bayes factor for the two must be sufficiently small or large. The tolerance of VoroInt can be adjusted to prioritize either time-efficiency or accuracy and reliability. Therefore, the best approach may be to do the following:

1. Compute a preliminary evidence for each model using VoroInt with a large tolerance (and, therefore, a short runtime).
2. Continue to compute the evidence for each model with a relatively large preliminary evidence using VoroInt with a small tolerance (and, therefore, a long runtime).
3. Select a model if possible.

Doing so would reduce the total runtime, assuming the evidences are computed sequentially and some evidence computation is discontinued. Moreover, because the cost of an insertion increases as the dimensionality increases, so does the reduction of the total runtime.

In applications in which an evaluation is extremely expensive (for example, it

is a physical experiment), the evaluation-efficiency of the method used to compute an integral is most important. Therefore, in such applications, VoroInt will almost certainly excel.

6.1 Future Work

A comparison of VoroInt to (Cosmo)PMC ([33-36]), which is an adaptive importance sampling method like VEGAS, may be worthwhile because VEGAS excels on certain integrands. A recomparison to each of the Monte Carlo methods using quasi-random sequences may also be worthwhile because the resulting Quasi-Monte Carlo methods converge more quickly than their counterparts ([37]). A comparison to recent implementations of nested sampling, such as PolyChord ([38-39]), may be worthwhile because another prescription for sampling, such as the slice sampling technique that PolyChord utilizes, may be better than the prescription that MultiNest utilizes. The inclusion of results that are averaged over several runs of the methods would instill confidence that the results that are included are typical.

VoroInt could estimate the area of a region without computational geometry by way of rejection sampling from the bounding box of the region, which was not done to limit the sources of the error in the integral. Moreover, VoroInt may be able to construct the bounding box of a region from the vertices of the local decomposition that is used to insert a sample into the global decomposition, which was not attempted for simplicity and because, for large d, most of the regions probably require additional vertices to bound them. VoroInt also may be able to directly determine the additional vertices if it were given or were to reconstruct the facets that define the explicit decomposition. However, directly determining the additional vertices may be complex.

The bias in the integral of VoroInt, which is caused by sites being on the low/high side of their regions where f is concave down/up, could be reduced by

using alternative values that are nearer the average values over the regions than the values at the sites in the regions to compute the integral. The alternative values could be the interpolated values at the centroids of the regions, the average of some interpolated values in the regions, and so on. If alternative values were used, then the error computed by VoroInt should be accordingly modified. The bias could also be reduced by using alternative values to compute the sampling priority. However, bias reduction by way of sampling probably would adversely affect the sampling efficiency when there are many regions over which the integral is biased.

The premature-termination guard that is employed by VoroInt and manually specified, could be automatically inferred from the distribution of the sampling priorities. The skewness of the distribution should become (more) negative as the highest-priority regions are sampled and, thereby, indicate the maturity of the sampling process. If the guard were not manually specified, then a user of VoroInt would only need to specify a maximum number of samples or a tolerance.

VoroPack, a package of Voronoi methods that operate on the piecewise-constant approximation of f constructed by VoroInt, the integrator in the package, is being developed. Some of the methods in the package will be:

- VoroOpt, the optimizer, which will minimize/maximize f by repeatedly sampling the piece (that is, Voronoi region) with the lowest/highest value and updating the approximation (that is, Voronoi decomposition).
- VoroInterp, the interpolator, which will approximate the value of f at an arbitrary point in D by way of nearest- or natural-neighbor interpolation or inverse-distance weighting of the natural neighbors of variable degree.
- VoroDiff, the differentiator, which will approximate the rate of change of f at an arbitrary point in D in an arbitrary direction by way of finite differences of evaluated or interpolated values.
- VoroSamp, the sampler, which will approximate a fair sample of evaluated or interpolated values of f from D by sampling each piece of f

with a frequency that is proportional to the approximate integral of f over the piece (that is, the area of the piece times the constant value of the piece).

VoroSamp would be an attractive alternative to MCMC methods because, unlike them, VoroSamp would not want correlation reduction or a burn-in period or need a proposal distribution. Moreover, VoroSamp could be used for parameter estimation and, therefore, would protect VoroInt from being criticized for not solving the full Bayesian inference problem as [40] was in [41]. Incidentally, the idea of using Voronoi integration (but ignoring unbounded regions) to compute Bayesian evidence is mentioned in [40] but MCMC methods are used instead. If VoroSamp were to update the approximation, then the samples in the approximation would approach a fair sample. If also $f \equiv 1$, then the samples would approach the uniform distribution.

The neighborhoods of the sites have properties that can be exploited by VoroInt or its extensions. For example, if a site were lower/higher than its neighbors, then it would be the local minimum/maximum. VoroInt could report the locations of the local extrema and, subsequently, VoroOpt could operate on the approximations of the neighborhoods of the extrema to do multi-objective optimization. The concavity of f about the extrema could be estimated by fitting a paraboloid that is centered at an extremum through the neighbors of the extremum. Other properties may emerge from the extended neighborhoods (that is, neighbors of neighbors) of the sites.

References

1. Cosmological Model Selection. Andrew R. Liddle, Pia Mukherjee, and David Parkinson. Astronomy & Geophysics, 47, 4.30-4.33, 2006.
2. Parallel Algorithms for Globally Adaptive Quadrature. J. Mark Bull. University of Manchester, 1997.
3. The Monte Carlo Method. Nicholas Metropolis, and S. Ulam. Journal of the American Statistical Association, Volume 44, Number 247, Pages 335-341, 1949.
4. Equation of State Calculations by Fast Computing Machines. Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Journal of Chemical Physics, 21, 1087, 1953.
5. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. W. K. Hastings. Biometrika, Volume 57, Number 1, Pages 97-109, 1970.
6. Algorithm 145: Adaptive Numerical Integration by Simpson's Rule. William Marshall McKeeman. Communications of the ACM, Volume 5 Issue 12, Page 604, 1962.
7. Qhull: http://www.qhull.org/
8. The Quickhull Algorithm for Convex Hulls. C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. Association for Computing Machinery Transactions on Mathematical Software, Volume 22 Issue 4 Pages 469-483, 1996.
9. Voronoi Diagrams from Convex Hulls. Kevin Q.Brown. Information Processing Letters, Volume 9, Issue 5, Pages 223-228, 1979.
10. GNU Scientific Library: https://www.gnu.org/software/gsl/
11. A New Algorithm for Adaptive Multidimensional Integration. G. P. Lepage. Journal of Computational Physics, 27, 192-203, 1978.
12. VEGAS: An Adaptive Multi-Dimensional Integration Program. G. P. Lepage. Cornell University Laboratory of Nuclear Studies, 447, 1980.
13. Recursive Stratified Sampling for Multidimensional Monte Carlo Integration. W. H. Press, and G. R. Farrar. Computers in Physics, 4,

190-195, 1990.

14. Nested Sampling. John Skilling. American Institute of Physics Conference Proceedings, 735, 395-405, 2004.
15. Nested Sampling for General Bayesian Computation. John Skilling. Maximum Entropy Data Consultants Limited, 2005.
16. Nested Sampling for General Bayesian Computation. John Skilling. International Society for Bayesian Analysis, 1, 4, 833-860, 2006.
17. Nested Sampling for Bayesian Computations. John Skilling. Proceedings Valencia / ISBA 8th World Meeting on Bayesian Statistics, 2006.
18. Discussion of Nested Sampling for Bayesian Computations by John Skilling. Michael J. Evans. Proceedings Valencia / ISBA 8th World Meeting on Bayesian Statistics, 2006.
19. Data Analysis: A Bayesian Tutorial / Second Edition. D. S. Sivia with J. Skilling. Oxford University Press, 2006.
20. Nested Sampling's Convergence. John Skilling. Maximum Entropy Data Consultants Limited, 2008.
21. Nest: http://www.inference.org.uk/bayesys/
22. MultiNest: https://ccpforge.cse.rl.ac.uk/gf/project/multinest/
23. A Nested Sampling Algorithm for Cosmological Model Selection. Pia Mukherjee, David Parkinson, and Andrew R. Liddle. Astrophysical Journal, 638, L51, 2006.
24. A Bayesian Model Selection Analysis of WMAP3. David Parkinson, Pia Mukherjee, and Andrew R. Liddle. Physical Review D, 73, 123523, 2006.
25. Efficient Bayesian Inference for Multimodal Problems in Cosmology. J. R. Shaw, M. Bridges, and M. P. Hobson. Monthly Notices of the Royal Astronomical Society 378, 1365-1370, 2007.
26. Multimodal Nested Sampling: An Efficient and Robust Alternative to MCMC Methods for Astronomical Data Analysis. F. Feroz, and M. P. Hobson. Monthly Notices of the Royal Astronomical Society 384, 449-463, 2008.
27. MultiNest: An Efficient and Robust Bayesian Inference Tool for

Cosmology and Particle Physics. F. Feroz, M. P. Hobson, and M. Bridges. Monthly Notices of the Royal Astronomical Society 398, 1601-1614, 2009.

28. Importance Nested Sampling and the MultiNest Algorithm. F. Feroz, M. P. Hobson, E. Cameron, and A. N. Pettitt. arXiv:1306.2144v2 [astro-ph.IM], 2014.
29. VORONOI_WEIGHT: http://people.sc.fsu.edu/~jburkardt/f_src/voronoi_weight/voronoi_weight.html
30. High-Performance Computation of Distributed-Memory Parallel 3D Voronoi and Delaunay Tessellation. Tom Peterka, Dmitriy Morozov, and Carolyn Phillips. SC14 (Supercomputing Conference), 2014.
31. https://www.wolframalpha.com/ Input: Integrate[Exp[-0.5((Sqrt[x^2+y^2]-1/3)/0.05)^2],{x,-0.5,0.5},{y,-0.5,0.5}]
32. Bayesian Evidence for a Cosmological Constant Using New High-Redshift Supernova Data. Paolo Serra, Alan Heavens, and Alessandro Melchiorri. Monthly Notices of the Royal Astronomical Society 379, 169-175, 2007.
33. Adaptive Importance Sampling in General Mixture Classes. Olivier Cappé, Randal Douc, Arnaud Guillin, Jean-Michel Marin, and Christian P. Robert. Statistics and Computing, Volume 18, Issue 4, Pages 447–459, 2008.
34. Estimation of cosmological parameters using adaptive importance sampling. Darren Wraith, Martin Kilbinger, Karim Benabed, Olivier Cappé, Jean-François Cardoso, Gersende Fort, Simon Prunet, and Christian P. Robert. Physical Review D, 80, 023507, 2009.
35. Bayesian model comparison in cosmology with Population Monte Carlo. Martin Kilbinger, Darren Wraith, Christian P. Robert, Karim Benabed, Olivier Cappé, Jean-François Cardoso, Gersende Fort, Simon Prunet, François R. Bouchet. Monthly Notices of the Royal Astronomical Society, Volume 405, Issue 4, Pages 2381–2390, 2010.
36. CosmoPMC: Cosmology Population Monte Carlo. Martin Kilbinger, Karim Benabed, Olivier Cappé, Jean Coupon, Jean-François Cardoso,

Gersende Fort, Henry J. McCracken, Simon Prunet, Christian P. Robert, and Darren Wraith. arXiv:1101.0950v3 [astro-ph.CO], 2012.

37. Quasi-Monte Carlo Methods and Pseudo-Random Numbers. Harald Niderreiter. Bulletin of the American Mathematical Society, Volume 84, Number 6, 1978.
38. PolyChord: Nested Sampling for Cosmology. W. J. Handley, M. P. Hobson, and A. N. Lasenby. Monthly Notices of the Royal Astronomical Society 000, 1-5, 2014.
39. PolyChord: Next-Generation Nested Sampling. W. J. Handley, M. P. Hobson, and A. N. Lasenby. Monthly Notices of the Royal Astronomical Society 000, 1-15, 2015.
40. Bayesian Evidence: Can We Beat MultiNest Using Traditional MCMC Methods? Rutger van Haasteren. arXiv:0911.2150v1 [astro-ph.IM], 2009.
41. Comment on “Bayesian Evidence: Can We Beat MultiNest Using Traditional MCMC Methods”, by Rutger van Haasteren (arXiv:0911.2150). F. Feroz, M. P. Hobson, and R. Trotta. arXiv:1001.0719v2 [astro-ph.IM], 2010.